This question paper contains (1) no. of printed page.

JNIT

JAGANNATH GUPTA INSTITUTE OF ENGINEERING & TECHNOLOGY JAIPUR

I-Mid Term Examination Session 2017-2018

B.Tech II Year IV Semester

Branch: CS | **Subject: MP&I**

Time: 10:00 to 11:30AM | **Subject Code: 4CS1A**

Date: 06/03/2018 | **Max. Marks: 20**

Note: Attempt any four questions out of five questions.

1. Explain the hardware model of 8085 microprocessor. 5

The hardware model in fig shows two major segments. One segment includes arithmetic logic unit [ALU] and an 8 bit register called an accumulator, instruction decoder, and flags. The flag registers are Zero, Carry, Sign, Parity and Auxilliary Carry flags. The second segment shows 8 bit and 16 bit registers. Both segments are connected with various internal connections called an internal bus. The arithmetic and logic operations are performed in the arithmetic logic unit [ALU]. Results are stored in the accumulator, and flip-flops, called flags, are set or reset to reflect the results. There are 3 buses- a 16 bit unidirectional address bus(A_0 TO A_{15}), an 8 bit bidirectional data bus(D_O TO D_7), and a control bus.

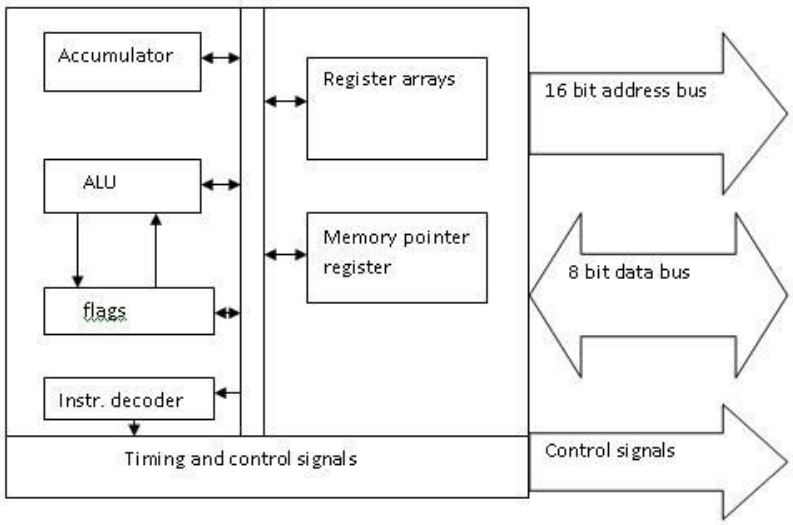
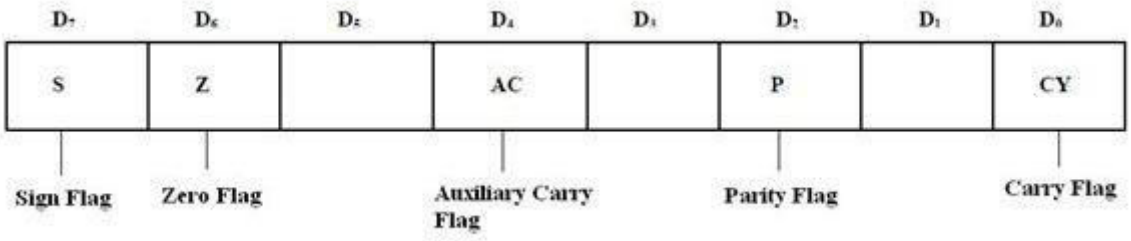


Fig. hardware model

2. Describe the different types of flag registers. 5

A flag is a flip flop. It indicates some condition produced by the execution of an instruction. The flag register of 8085 microprocessor consists of 5 flags. The flag register is connected to ALU. When an operation is performed by ALU the result is transferred on data bus and status of result will be stored in flip flops. The different flags and their positions in flag register are shown in following.



a) **The carry flag(CY):**

This flag is set whenever there has been a carry out of, or a borrow into, the higher order bit of the result. The flag is used by the instructions that add amd subtract instruction.

1-carry out from MSB bit on addition or borrow into MSB bit on subtraction

0-no carry out or borrow into MSB bit

b) **The parity flag(P)-**

This flag is set whenever the result has even parity, an even number of 1 bits. If parity is odd, PF is cleared.

1-low byte has even number of 1 bits

0-low byte has odd parity

c) **The auxiliary carry flag(AC):**

This flag is set whenever there has been a carry out of the lower nibble into the higher nibble or a borrow from higher nibble into the lower nibble of an 8 bit quantity, else AF is reset. This flag is used by decimal arithmetic instructions.

1-carry out from bit 3 on addition or borrow into bit 3 on subtraction

0-otherwise

d) **The zero flag(Z):**

This flag is set, when the result of operation is zero, else it is reset.

1-zero result

0-non-zero result

e) **The sign flag(S):**

This flag is set, when MSB (Most Significant Bit) of the result is 1. Since negative binary numbers are represented in the 8085 CPU in standard two's complement notation, S indicates sign of the result.

1-MSB is 1 (negative)

0-MSB is 0 (positive)

3. Write an assembly level language program for 8 bit add operation 5

Here, the HL register pair is first initialized to the start address of memory at which the data is stored. Then data is brought to accumulator A and the other one is added from memory itself. The result from A is then stored into memory again using the HL register.

LXI H,3000

MOV A,M

INX H

ADD M

INX H

MOV M,A

HLT

The given data are present at memory locations 3000_H and 3001_H and the result is stored at memory location 3002_H

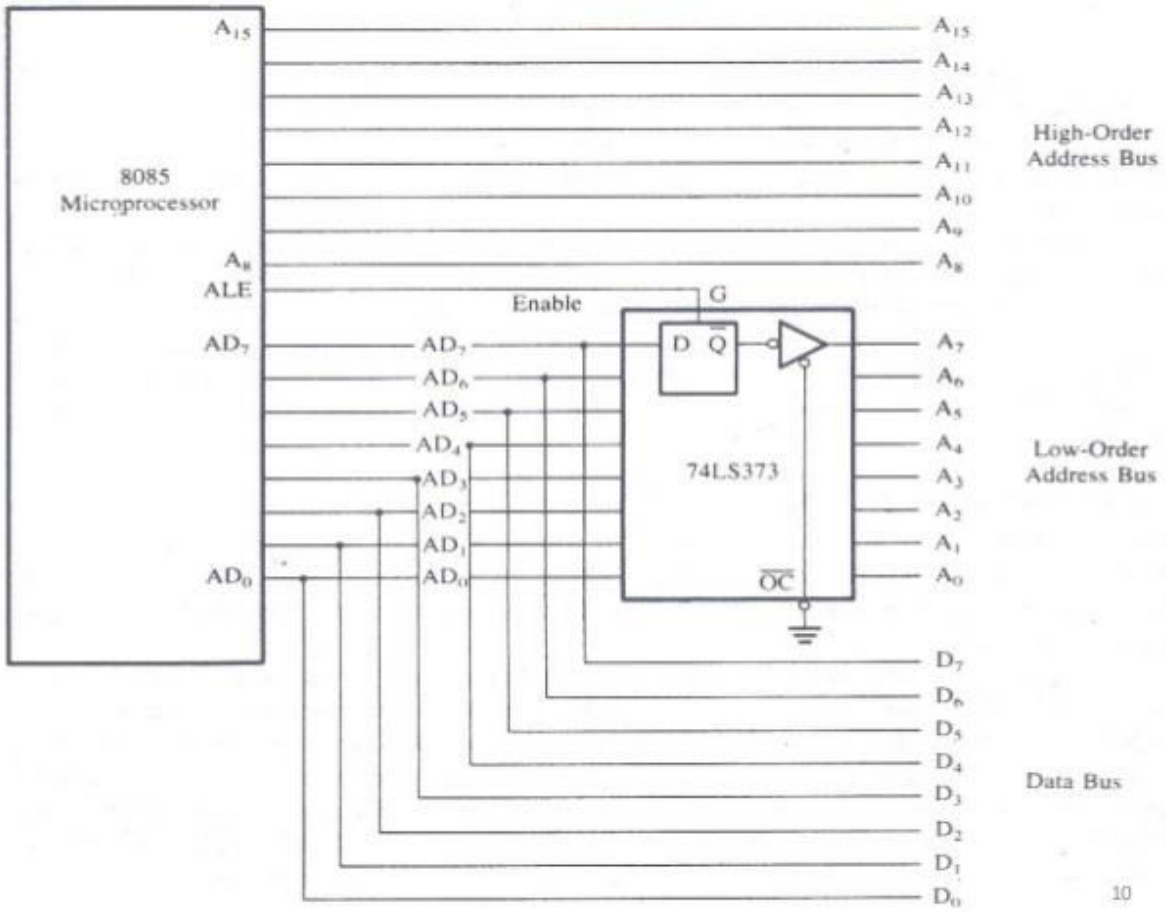
4. Explain the demultiplexing process of address and data buses in 8085. 5

Whenever an instruction is executed by MPU first of all MPU sends ALE signal to address latch IC to enable all D Latches to receive new address from MPU.

In first T state Microprocessor generates the address on Address Bus, half portion of address (lower order address) is generated on AD0-AD7 . This Address bits are captured by D latches and stored in.

During next cycles say T2, T3 and so on, MP can use AD0-AD7 as Data Bus to send receives data. During this period the initially generated Address is also available at output

pins of D Latches .The IC 74LS373is used as Address Latch it contains 8 D Latches to store lower half of address.(8 bits).



5 Write short notes on: 5

(a) Stack Pointer:

A stack pointer is a small register that stores the address of the last program request in a stack. The Stack pointer is a sixteen bit register used to point at the stack.In read write memory the locations at which temporary data and return addresses are stored is known as the stack.In simple words stack acts like an auto decrement facility in the system.The initialization of the stack top is done with the help of an instruction LXI SP. In order to avoid program crashes a program should always be written at one end and initialized at the other.

(b) LDAX:

LDAX(Load accumulator indirect): The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. The contents of either the register pair or the memory location are not altered.

LDAX B

(c) LHLD

LHLD(Load H and L register direct): - this instruction loads the contents of the 16- bit memory location into the HL register pair.

LHLD 3000H (the content of location 3000h is copied into the HL reg pair

(d) STA

STA: the content of accumulator are copied into the memory location.

STA 3000H (the content of accumulator is stored into the memory location 3000h)

(e) XRA

The content of accumulator are exclusive OR with specified register or memory location.

. XRA B :ExOR register B with accumulator

XRA M :ExOR data pointed to by HL pair with accumulator.

JNIT

JAGANNATH GUPTA INSTITUTE OF ENGINEERING & TECHNOLOGY JAIPUR

I / II -MID TERM PAPER ANSWER SHEET

Semester: IV

Subject : DMS

Branch : CS

Submitted by : Ms. Daljeet Kaur

1. Statement :- Let n- pigeons are assigned R pigeonholes, then one of the pigeonholes must contain at least $\left\lfloor \frac{n-1}{R} \right\rfloor + 1$ pigeons, where $\left\lfloor \frac{n-1}{R} \right\rfloor$ is the floor of $\frac{n-1}{R}$.

Proof! We have n- pigeons and R- pigeonholes such that $n > R$. Assuming that each of R pigeonhole contains not more than $\left\lfloor \frac{n-1}{R} \right\rfloor$ pigeons, then total number of pigeons in the R pigeonholes must be less than or equal to

$$R \times \left\lfloor \frac{n-1}{R} \right\rfloor \leq R \times \frac{n-1}{R} = n-1$$

but there are n- pigeons, so this contradicts our assumptions that a pigeonhole contains not more than $\left\lfloor \frac{n-1}{R} \right\rfloor$ pigeons. So, one of the pigeonhole must contain at least $\left\lfloor \frac{n-1}{R} \right\rfloor + 1$ pigeons.

2. Assuming $f(x)$ to be real number, domain will consists of those values of x for which :

(a) $f(x)$ does not tend to ∞.

(b) $f(x)$ does not becoming involving $\sqrt{-1}$

(c) $f(x)$ does not become indeterminate

(i) $f(x) = \frac{x}{x^2+1}$ is well defined $\forall x \in R$.

So domain of $f(x)$ = set of real number R.

(ii) $g(x) = \frac{x}{x-5} \longrightarrow \infty$ for $x = 5$

So domain $g(x) = R - \{5\}$

Contd....

(3) To prove $A \times (B \cup C) = (A \times B) \cup (A \times C)$

we have to show that (i) $A \times (B \cup C) \subseteq (A \times B) \cup (A \times C)$

(ii) $(A \times B) \cup (A \times C) \subseteq A \times (B \cup C)$

(i) Let $(x, y) \in A \times (B \cup C) \Rightarrow x \in A$ and $y \in B \cup C$

$\Rightarrow x \in A$ and $y \in B$ or $y \in C$

Therefore, either $x \in A$ and $y \in B$ or $x \in A$ and $y \in C$

$\Rightarrow (x, y) \in A \times B$ or $(x, y) \in A \times C$

$\Rightarrow (x, y) \in (A \times B) \cup (A \times C)$

So $A \times (B \cup C) \subseteq (A \times B) \cup (A \times C)$

(ii) Now, let $(x, y) \in (A \times B) \cup (A \times C)$

$\Rightarrow (x, y) \in (A \times B)$ or $(x, y) \in (A \times C)$

$\Rightarrow (x \in A, y \in B)$ or $(x \in A, y \in C)$

It means $x \in A$ always & y may $\in B$ or C.

So $x \in A$ and $y \in B$ or $y \in C$

$\Rightarrow x \in A$ and $y \in (B \cup C)$

$\Rightarrow (x, y) \in A \times (B \cup C)$

so $(A \times B) \cup (A \times C) \subseteq A \times (B \cup C)$

From (i) and (ii) $\Rightarrow A \times (B \cup C) = (A \times B) \cup (A \times C)$

(9) Given $A = \{a, b, c, d, e\}$ and $B = \{c, e, f, h, k, m\}$

Then $A \cup B = \{a, b, c, d, e, f, h, k, m\}$

$A \cap B = \{c, e\}$

$\therefore |A| = 5$, $|B| = 6$, $|A \cup B| = 9$ $|A \cap B| = 2$

L.H.S. $= |A \cup B| = 9$

R.H.S. $= |A| + |B| - |A \cap B| = 5 + 6 - 2 = 9$

$\therefore$ L.H.S = R.H.S.

Hence $|A \cup B| = |A| + |B| - |A \cap B|$

contd...

(5) Let O be the set of odd positive integers and N be the set of natural numbers.

Consider a function $f : N \to O$

such that $f(n) = 2n-1$, $n \in N$

We need to show that f is bijection.

One-one : Let $f(n_1) = f(n_2)$

$\Rightarrow 2n_1 - 1 = 2n_2 - 1$

$\Rightarrow 2n_1 = 2n_2$

$\Rightarrow n_1 = n_2$

$\therefore$ f is one-one

Onto : Let $y \in O$. then y is an odd positive integer

$\Rightarrow y+1$ is an even positive integer

$\Rightarrow \frac{y+1}{2}$ is a positive integer

$\Rightarrow \frac{y+1}{2} \in N$

Now for each $y \in O$ $\exists \left(\frac{y+1}{2}\right) \in N$ such that

$$f\left(\frac{y+1}{2}\right) = 2\left(\frac{y+1}{2}\right) - 1 = (y+1) - 1 = y$$

so each element of O has its preimage in N. Thus f is onto.

Hence f is a bijection between N and O.

$|N| = |O|$

$\therefore$ O is countable,

ie. the set of odd positive integers is a countable set.

JNIT

JAGANNATH GUPTA INSTITUTE OF ENGINEERING & TECHNOLOGY JAIPUR
VI / II -MID TERM PAPER ANSWER SHEET

Semester: IV
Branch : Computer
Subject : SPT.
Submitted by : Chandresh Mathur

Q.1 State and Prove Bay's theorem

Ans:- Statement — If $E_1, E_2 \ldots E_n$ be n mutually exclusive and exhaustive Set of events of a sample space S and E is an event which occure together with either of $E_1, E_2, \ldots E_n$ (form a partition of S) and Let E be any event then

$$P(E_i/E) = \frac{P(E_i)\,P(E/E_i)}{\sum_{j=1}^{n} P(E_j)\,P(E/E_j)}$$

Proof: Since $E_1, E_2, \ldots E_n$ be the Partition of S.

Hence $S = E_1 \cup E_2 \cup E_3 \cup \ldots \cup E_n$ and $E_i \cap E_j = \phi \quad i \neq j$.

Now $E = E \cap S = E \cap [E_1 \cup E_2 \cup E_3 \cup \ldots \cup E_n]$

$= (E \cap E_1) \cup (E \cap E_2) \cup (E \cap E_3) \cup \ldots \cup (E \cap E_n)$ (Distributive Prop.)

$P(E) = P(E \cap E_1) + P(E \cap E_2) + P(E \cap E_3) + \ldots + P(E \cap E_n)$ (By. Add. Prop.)

$$P(E_i/E) = \frac{P(E \cap E_i)}{P(E)} \quad \text{(Conditional Prob.)}$$

$$= \frac{P(E \cap E_i)}{P(E \cap E_1) + P(E \cap E_2) + P(E \cap E_3) + \ldots + P(E \cap E_n)}$$

$$= \frac{P(E_i)\,P(E/E_i)}{P(E_1)\,P(E/E_1) + P(E_2)P(E/E_2) + \ldots + P(E_n)\,P(E/E_n)}$$

$$= \frac{P(E_i)\,P(E/E_i)}{\sum_{i=1}^{n} P(E_1)\,P(E/E_i)}$$

Q.2 Three Factories A, B, C does 30%, 50%, and 20% production of certain item. out of their production 8%, 5%, 10% of the item produced are defective respectively. an item is purchased and is found to be defective. Find the probability that it was a product of factory A.

Ans :- Let Events be

$E_1 \rightarrow$ Item is Produced by A.

$E_2 \rightarrow$ Item is Produced by B

$E_3 \rightarrow$ Item is Produced by C

$E \rightarrow$ Item Purchased is defective

It is given that

$P(E_1) = \frac{3}{10}$ $\quad P(D/E_1) = \frac{8}{100}$

$P(E_2) = \frac{5}{10}$ $\quad P(D/E_2) = \frac{5}{100}$

$P(E_3) = \frac{2}{10}$ $\quad P(D/E_3) = \frac{10}{100}$

Now By Bays theorem

$$P(E_1/D) = \frac{P(E_1)\,P(D/E_1)}{P(E_1)\,P(D/E_1) + P(E_2)\,P(D/E_2) + P(E_3)\,P(D/E_3)}$$

$$= \frac{3/10 \cdot 8/100}{\frac{3}{10}\times\frac{8}{100} + \frac{5}{10}\cdot\frac{5}{100} + \frac{2}{10}\cdot\frac{10}{100}} = \frac{24/1000}{\frac{24}{1000}+\frac{25}{1000}+\frac{20}{1000}}$$

$$= \frac{24}{69} = \frac{8}{23} \quad \text{Ans}$$

Q.3 Probability of getting 9 on two dice

$P(9) = \frac{4}{36} = \frac{1}{9}$ let it call sucess and represent by p

$p = \frac{1}{9} =$ Probability of getting 9

$q = 1-p = \frac{8}{9}$ Probability of not getting 9

When A start the game he can win in, 1, 3, 5, 7, 9.. chance

i.e chance of wining $A = p + q^2p + q^4p + q^6p + \dots = \frac{p}{1-q}$

$= \frac{p}{1-q^2}$ [G.P. of infinite term]

$= \frac{1/9}{1-(8/9)^2} = \frac{1/9}{81-64/81} = \frac{9}{17}$

Cont: Chance of wining B is in 2, 4, 6, 8, 10, -- chance.

Hence the chance of wining B is

$$= qp + q^3p + q^5p + q^7p + \cdots \infty$$

= It is also a G.P. (with q^2 common ratio)

$$= \frac{qp}{1-q^2} = \frac{8/9 \cdot 1/9}{1-(8/9)^2} = \frac{8/81}{1-64/81}$$

$$= \frac{8/81}{17/81} = \frac{8}{17}$$

Chance of wining A = 9/17

" " B = 8/17

Chance of wining Ratio. A : B = 9 : 8.

Q.4 Given $P(A) = 3/8$ $P(B) = 5/8$ $P(A \cup B) = 3/4$.

Find. (i) $P(\bar{A} \cap B)$ (ii) $P(\bar{B} \cap A)$ (iii) $P(A/B)$

Ans We know

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

$$3/4 = \frac{3}{8} + \frac{5}{8} - P(A \cap B)$$

$$P(A \cap B) = 1 - \frac{3}{4} = \frac{1}{4}$$

(i) $P(\bar{A} \cap B) = P(B) - P(A \cap B) = \frac{5}{8} - \frac{1}{4} = \frac{3}{8}$

(ii) $P(A \cap \bar{B}) = P(A) - P(A \cap B) = \frac{3}{8} - \frac{1}{4} = \frac{1}{8}$

(iii) $P(A/B) = \frac{P(A \cap B)}{P(B)} = \frac{1/4}{5/8} = \frac{1}{4} \times \frac{8}{5} = \frac{2}{5}$

Q.5 A Random Variable X has the following Prob. distribution

X	0	1	2	3	4	5	6	7
$P(x)$	0	k	$2k$	$2k$	$3k$	k^2	$2k^2$	$7k^2+k$

(i) Find the value of k

(ii) $P\left(\frac{1.5 < X < 4.5}{X > 2}\right)$

(iii) The smallest value λ for which $P(X \leq \lambda) > \frac{1}{2}$

Ans :- We know By the definition of PMF $\sum_i P_i = 1$

(i) Hence $0+k+2k+2k+3k+k^2+2k^2+7k^2+k = 1$

$$10k^2+9k = 1$$

$$(10k-1)(k+1) = 0 \quad \Rightarrow \quad k = -1 \text{ or } \frac{1}{10}$$

$k = -1$ Not Possible as $P_i \geq 0$

$k = 1/10$

(ii) $$P\left(\frac{1.5 < X < 4.5}{X > 2}\right) = P\left(\frac{(1.5 < x < 4.5) \cap (X > 2)}{P(X > 2)}\right)$$

$$= \frac{P(2 < X < 4.5)}{P(X > 2)} = \frac{P(3)+P(4)}{1 - P(0) - P(1) - P(2)}$$

$$= \frac{\frac{2}{10}+\frac{3}{10}}{1-\left\{0+\frac{1}{10}+\frac{2}{10}\right\}} = \frac{\frac{5}{10}}{1-\frac{3}{10}} = \frac{5/10}{7/10}$$

$$= \frac{5}{7}$$

(iii) $$P(X \leq 3) = P(0)+P(1)+P(2)+P(3) = \frac{5}{10} = \frac{1}{2}$$

$$P(X \leq 4) = P(0)+P(1)+P(2)+P(3)+P(4) = \frac{8}{10} = \frac{4}{5} > \frac{1}{2}$$

So Smallest of λ satisfying the condition

$$P(X \leq \lambda) > \frac{1}{2}$$

is $\underline{4}$ Ans.

Q.6 The Joint Probability distribution of Random variable (x, y) is

$P_{xy}(x_i, y_j) = \frac{1}{27}(x_i + 2y_j)$ where x and y can assume only the integer values $0, 1, 2$

Find (i) Marginal distribution of x and y

(ii) Are x & y are independent

(iii) Find the conditional distribution of y for $x = 2$

Ans: As (X, Y) is a bivariate random variable. then in tabular form $P_{xy} = \frac{1}{27}(x_i + 2y_j)$ can represent as follows.

X \ Y	0	1	2	Total
0	0	2/27	4/27	6/27
1	1/27	3/27	5/27	9/27
2	2/27	4/27	6/27	12/27
Total	3/27	9/27	15/27	1

(i) The Marginal Probability Mass function of X

$$P_x(x_i) = \sum_j P_{xy}(x_i, y_j) = \sum_{j=0}^{2} P_{xy}(x_i, y_j)$$

X=i	$P_x(x_i)$
0	6/27
1	9/27
2	12/27

Marginal Prob. Mass function of Y.

$$P_y(y_j) = \sum_i P_{xy}(x_i, y_j) = \sum_{i=0}^{2} P_{xy}(x_i, y_i)$$

Y=i	0	1	2
$P_y(y_j)$	3/27	9/27	15/27

(ii) Two Random variable are called independent if

$$P_{xy}(x_i, y_j) = P_x(x_i) \cdot P_y(y_j)$$

$P_{xy}(0,0) = 0$ but $P_x(0) = 6/27$ $P_y(0) = \frac{3}{27}$

$0 \neq \frac{6}{27} \times \frac{3}{27}$ $\Rightarrow$ Hence two variables are not independent

(iii) Conditional distribution of y for $x = 2$

$$P_{\frac{y}{x}}\left(\frac{y_j}{x_i}\right) = \frac{P_{xy}(x_i, y_j)}{P_x(x_i = 2)} =$$

y_i	0	1	2
$P_{y/x}$	$\frac{2/27}{12/27} = 1/6$	$\frac{4/27}{12/27} = 1/3$	$\frac{6/27}{12/27} = 1/2$

Q.6b

According to question let x and y denotes the Number of red and white ball choosen. So x and y taken values $0,1,2$ subject to condition $x+y \geq 0$ the total No. of balls. $= 9$

No. of ways to choose two balls

$$9C_2 = \frac{9.8}{2.1} = 36$$

X \ Y	0	1	2	Total
0	1/6	1/3	1/12	7/12
1	2/9	1/6	0	7/18
2	1/36	0	0	1/36
Total	15/36	1/2	1/12	1

(i) $P_{xy}(00) = \frac{3C_0\, 2C_0\, 4C_2}{9C_2} = \frac{1}{6}$

$P_x(0,1) = \frac{3C_0\, 2C_1\, 4C_1}{9C_2} = \frac{1}{3}$

$P(0,2) = \frac{3C_0\, 2C_2\, 4C_0}{9C_2} = \frac{1}{12}$

$P(1,0) = \frac{3C_1\, 2C_0\, 4C_1}{9C_2} = \frac{2}{9}$

$P(2,0) = \frac{3C_2\, 2C_0\, 4C_1}{9C_2} = \frac{1}{36}$

$P(1,1) = \frac{3C_1\, 2C_1\, 4C_0}{9C_2} = \frac{1}{6}$

(ii) Now Marginal distribution of x

$$P_x(x_j) = \sum_{j=0}^{2} P_{xy}(x_i y_j) = P(x_j 0) + P(x_j, 1) + P(x_j, 2)$$

$P_x(0) = P(0,0) + P(0,1) + P(0,2) = 7/12$

$P_x(1) = P(1,0) + P(1,1) + P(1,2) = 7/18$

$P_x(2) = P(2,0) + P(2,1) + P(2,2) = 1/36$

Marginal distribution of Y.

$$P_y(y_i) = \sum_{i=0}^{2} P_{xy}(x_i y_j) = P(0, y_j) + P(1, y_j) + P(2, y_j)$$

$P_y(0) = P(0,0) + P(1,0) + P(2,0) = 15/36$

$P_y(1) = P(0,1) + P(1,1) + P(2,1) = 1/2$

$P_y(2) = P(0,2) + P(1,2) + P(2,2) = 1/12$

(iii) Two variable are called independant if

$$P_{xy}(x_i y_j) = P_x(x_j) \times P_y(y_j)$$

But in this $P_x(0,0) = 1/6$ $\quad P_x(0) = \frac{7}{12}$ & $P_y(0) = \frac{15}{36}$

$$\frac{1}{6} \neq \frac{7}{12} \times \frac{5}{36}$$

Hence the variable are not independant

Q.1 Briefly explain various phases of system development life cycle (SDLC)?

Ans. System life cycle is an organizational process of developing and maintaining systems. It helps in establishing a system project plan, because it gives overall list of processes and sub-processes required for developing a system. System development life cycle means combination of various activities. In other words we can say that various activities put together are referred as system development life cycle. In the System Analysis and Design terminology, the system development life cycle also means software development life cycle. Following are the different phases of system development life cycle:

- Preliminary study
- Feasibility study
- Detailed system study
- System analysis
- System design
- Coding
- Testing

Let us now describe the different phases and related activities of system development life cycle.

(a) Preliminary System Study

Preliminary system study is the first stage of system development life cycle. This is a brief investigation of the system under consideration and gives a clear picture of what actually the physical system is? In practice, the initial system study involves the preparation of a 'System Proposal' which lists the Problem Definition, Objectives of the Study, Terms of reference for Study, Constraints, Expected benefits of the new system, etc. in the light of the user requirements. The system proposal is prepared by the System Analyst (who studies the system) and places it before the user management. The management may accept the proposal and the cycle proceeds to the next stage. The management may also reject the proposal or request some modifications in the proposal. In summary, we would say that system study phase passes through the following steps:

- problem identification and project initiation
- background analysis
- inference or findings (system proposal)

(b) Feasibility Study In case the system proposal is acceptable to the management, the next phase is to examine the feasibility of the system. The feasibility study is basically the test of the proposed system in the light of its workability, meeting user's requirements, effective use of resources and of course, the cost effectiveness. These are categorized as technical, operational, economic and schedule feasibility. The main goal of feasibility study is not to solve the problem but to achieve the scope. In the process of feasibility study, the cost and benefits are estimated with greater accuracy to find the Return on Investment (ROI). This also defines the resources needed to complete the detailed investigation. The result is a feasibility report submitted to the management. This may be accepted or accepted with modifications or rejected. The system cycle proceeds only if the management accepts it.

(c) Detailed System Study The detailed investigation of the system is carried out in accordance with the objectives of the proposed system. This involves detailed study of various operations performed by a system and their relationships within and outside the system. During this process, data are collected on the available files, decision points and transactions handled by the present system. Interviews, on-site observation and questionnaire are the tools used for detailed system study. Using the following steps it becomes easy to draw the exact boundary of the new system under consideration: l Keeping in view the problems and new requirements l Workout the pros and cons including new areas of the system All the data and the findings must be documented in the form of detailed data flow diagrams (DFDs), data dictionary, logical data structures and miniature specification. The main points to be discussed in this stage are: l Specification of what the new system is to accomplish based on the user requirements.

- Functional hierarchy showing the functions to be performed by the new system and their relationship with each other.
- Functional network, which are similar to function hierarchy but they highlight the functions which are common to more than one procedure.
- List of attributes of the entities – these are the data items which need to be held about each entity (record)

(d) System Analysis Systems analysis is a process of collecting factual data, understand the processes involved, identifying problems and recommending feasible suggestions for improving the system functioning. This involves studying the business processes, gathering operational data, understand the information flow, finding out bottlenecks and evolving solutions for overcoming the weaknesses of the system so as to achieve the organizational goals. System Analysis also includes subdividing of complex process involving the entire system, identification of data store and manual processes. The major objectives of systems analysis are to find answers for each business process: What is being done, How is it being done, Who is doing it, When is he doing it, Why is it being done and How can it be improved? It is more of a thinking process and involves the creative skills of the System Analyst. It attempts to give birth to a new efficient

system that satisfies the current needs of the user and has scope for future growth within the organizational constraints. The result of this process is a logical system design. Systems analysis is an iterative process that continues until a preferred and acceptable solution emerges.

(e) System Design

Based on the user requirements and the detailed analysis of the existing system, the new system must be designed. This is the phase of system designing. It is the most crucial phase in the developments of a system. The logical system design arrived at as a result of systems analysis is converted into physical system design. Normally, the design proceeds in two stages:

- Preliminary or General Design
- Structured or Detailed Design

Preliminary or General Design: In the preliminary or general design, the features of the new system are specified. The costs of implementing these features and the benefits to be derived are estimated. If the project is still considered to be feasible, we move to the detailed design stage.

Structured or Detailed Design: In the detailed design stage, computer oriented work begins in earnest. At this stage, the design of the system becomes more structured. Structure design is a blue print of a computer system solution to a given problem having the same components and inter-relationships among the same components as the original problem. Input, output, databases, forms, codification schemes and processing specifications are drawn up in detail. In the design stage, the programming language and the hardware and software platform in which the new system will run are also decided. There are several tools and techniques used for describing the system design of the system. These tools and techniques are:

- Flowchart
- Data flow diagram (DFD)
- Data dictionary
- Structured English
- Decision table
- Decision tree

Each of the above tools for designing will be discussed in detailed in the next lesson. The system design involves:

- Defining precisely the required system output
- ii. Determining the data requirement for producing the output
- iii. Determining the medium and format of files and databases
- iv. Devising processing methods and use of software to produce output
- v. Determine the methods of data capture and data input vi. Designing Input forms

- vii. Designing Codification Schemes
- viii. Detailed manual procedures ix. Documenting the Design

(f) Coding

The system design needs to be implemented to make it a workable system. This demands the coding of design into computer understandable language, i.e., programming language. This is also called the programming phase in which the programmer converts the pro- language. This is also called the programming phase in which the programmer converts the program specifications into computer instructions, which we refer to as programs. It is an important stage where the defined procedures are transformed into control specifications by the help of a computer language. The programs coordinate the data movements and control the entire process in a system. It is generally felt that the programs must be modular in nature. This helps in fast development, maintenance and future changes, if required. Using the test data following test run are carried out:

- Program test
- System test

(g) Testing

Before actually implementing the new system into operation, a test run of the system is done for removing the bugs, if any. It is an important phase of a successful system. After codifying the whole programs of the system, a test plan should be developed and run on a given set of test data. The output of the test run should match the expected results. Sometimes, system testing is considered a part of implementation process.

Program test:

When the programs have been coded, compiled and brought to working conditions, they must be individually tested with the prepared test data. Any undesirable happening must be noted and debugged (error corrections) System Test: After carrying out the program test for each of the programs of the system and errors removed, then system test is done. At this stage the test is done on actual data. The complete system is executed on the actual data. At each stage of the execution, the results or output of the system is analysed. During the result analysis, it may be found that the outputs are not matching the expected output of the system. In such case, the errors in the particular programs are identified and are fixed and further tested for the expected output. When it is ensured that the system is running error-free, the users are called with their own actual data so that the system could be shown running as per their requirements.

(h) Implementation After having the user acceptance of the new system developed, the implementation phase begins. Implementation is the stage of a project during which theory is turned into practice. The major steps involved in this phase are:

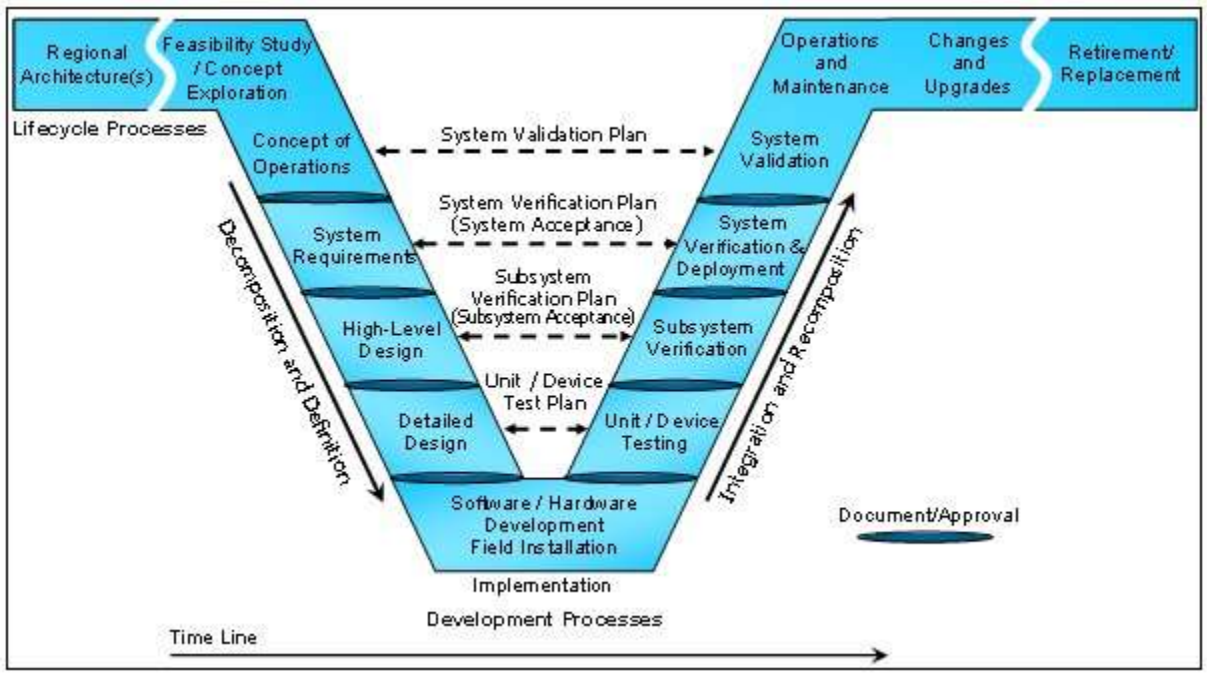
Acquisition and Installation of Hardware and Software

- Conversion
- User Training
- Documentation

The hardware and the relevant software required for running the system must be made fully operational before implementation. The conversion is also one of the most critical and expensive activities in the system development life cycle. The data from the old system needs to be converted to operate in the new format of the new system. The database needs to be setup with security and recovery procedures fully defined. During this phase, all the programs of the system are loaded onto the user's computer. After loading the system, training of the user starts. Main topics of such type of training are:

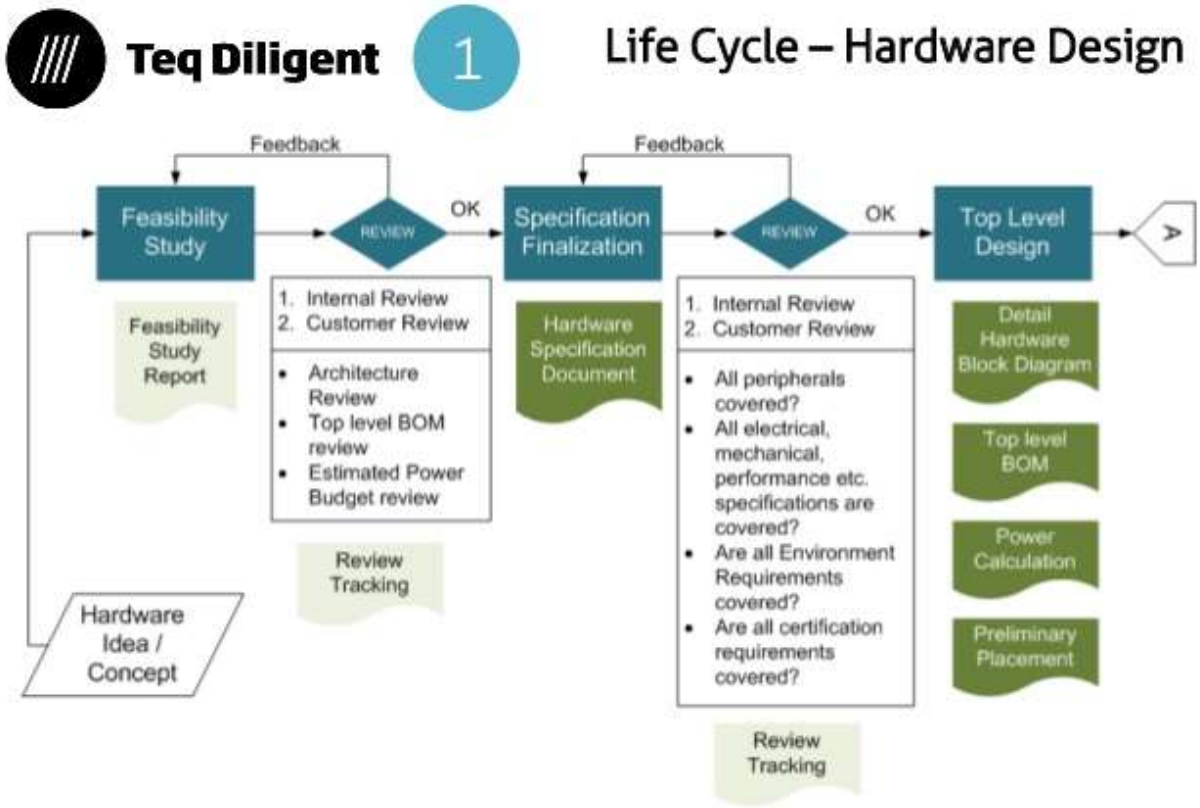
Q.2 Design and explain hardware engineering life cycle?

Answer: The systems engineering process (SEP) is a methodology and tool for managing a system's life cycle starting with concepts and ending with the system's retirement. It is a highly structured method to facilitate the development, maintenance, refinement, and retirement of dynamic, large-scale systems consisting of both technical components (Figure 2-1 is the Systems Engineering "V" Model for ITS that details the various stages that occur within the system's life cycle.



While testing is shown as **one** stage of the life cycle, it is important to understand that testing is also a **continuous process** within the life cycle. Testing begins with writing the requirements; each requirement must be written in a manner that allows it to be tested. During the design stages, testing will be a consideration as design trade-offs are evaluated for their ability to satisfy the requirements. New requirements may emerge from the designs as choices are made to satisfy the requirements within the project's constraints. Hardware components, software components, subsystems, and systems will be verified during the implementation and testing stages. Final system-level tests will be performed to accept the system and demonstrate the system's readiness for production service. However, testing activities will not end once the system is in operation; it will continue as the operations and maintenance staff perform corrective, adaptive, and other system maintenance activities.

The following sections will discuss these process activities in more detail with respect to the input information, selected processes, and the results from the testing process while verifying the transportation management system.



Note: All dark colour shaded phases/documents were part of Scope of this project.

Q.3 What do you mean by system analysis? Explain system analysis of existing system?

Ans: It is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components.

System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose.

Systems Analysis in Software Engineering

System analysis in software engineering is, therefore, the activities that comprise software engineering as a process in the production of software. It is the **software process**. This process has 4 main activities. They are:

- Software Specification
- Software Design and Implementation
- Software Validation
- Software Evolution.

As we can see, these activities are similar to those within systems analysis and the design of software. Depending on the methodology used, the activities can be arranged differently. They are arranged sequentially, for example, in the well-known Waterfall Model, while in the Incremental Development model they are inter-related.

Stages of Systems Analysis

Software Specification

This is also known as **requirements engineering** and is defined as the identification of the requirements of the system and the limitations within which the system will operate, develop or can evolve. This stage ensures that the software meets all the users' expectations. It ensures the delivery of quality software to the user at the end of the production process. On completion of the software specification, a requirements document will be produced and validated by all parties.

The requirements engineering stages are:

1. Feasibility studies

The user's needs are accessed to ensure that current technologies can adequately handle them, they are cost-effective, and they are within the limits of the overall budget. The feasibility study guides the ultimate decision as to whether to progress with the development or not.

2. Requirements Analysis

This involves stipulating system requirements from existing systems, potential users' inputs, and further analysis. Models are developed or decided on and the result ensures the system in question is properly understood.

3. Requirements Specification

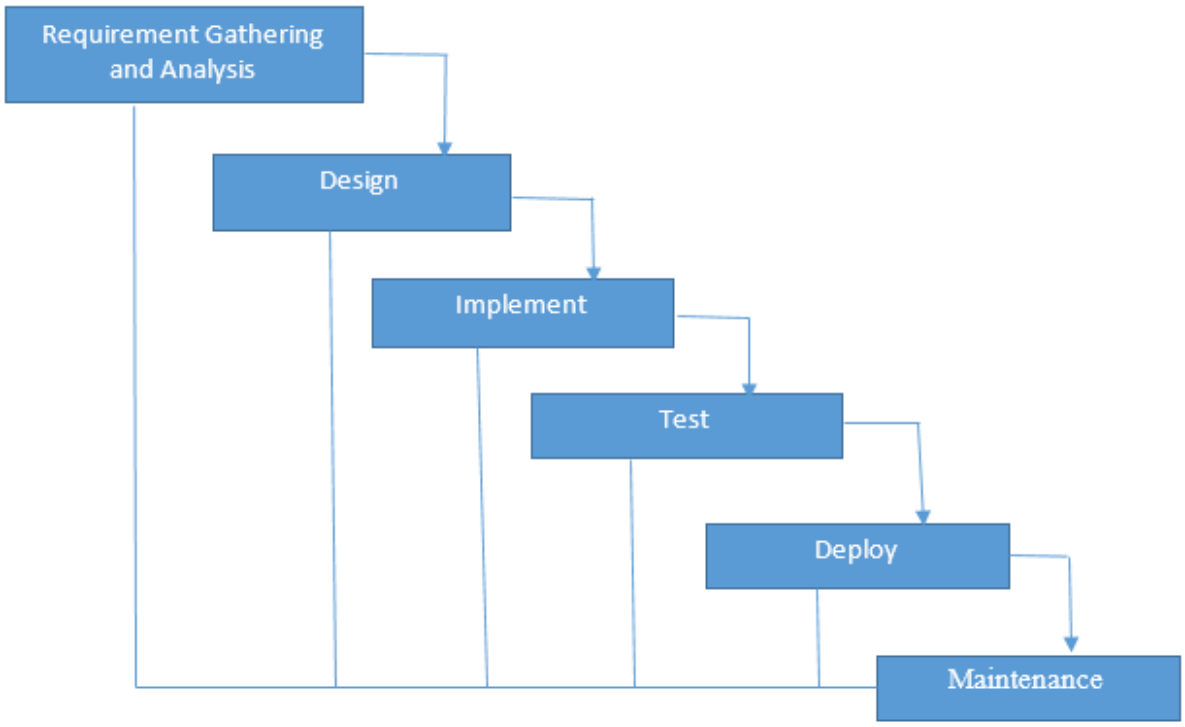
At this stage, all the information gathered so far is translated into a document that clearly states all the system requirements including the users' stated requirements and the detailed system functionalities.

4. Requirements Validation

This stage checks that the requirements developed are consistent and complete.

Q.4 Explain waterfall model using suitable diagram?

Ans.: Waterfall model is the classic Software Development Life Cycle method practiced in software development process. As the name "waterfall" describes, this development is flowing downwards steadily like waterfall, i.e., proceeds from one phase to another phase. This process is a sequential process or a linear process where in the output of one phase is input of another phase. So previous phase becomes important to be completed before we move to next phase. Here is the diagram that describes all phases of waterfall model.



Requirement Gathering and Analysis Phase: This is the first phase where requirements to produce a product are gathered and analyzed. This is the phase which involves customer. All information about the entire process are identified to be designed and delivered here.

Design Phase: The requirements from the earlier phase are documented and converted into technical design. Like what hardware, system software, technology, language are used etc. are specified.

Implement Phase: Output from Design phase are used and implemented to achieve the goal. They are split in program units. These program units are developed independently and functionally tested. This is called Unit Testing.

Test Phase: Here all program units which are developed in implement phase are integrated and tested together to see end product has all desired functionalities required.

Deploy Phase: Once Test phase is successfully completed, it is deployed in customer environments and product is released.

Maintenance Phase: If any changes are required in client environments then they are upgraded and released as patches to fix any issues that come up after deployment.

Waterfall Model Pros:

1. Simple, easy to understand and use.
2. All phases are clearly documented and understood well in the beginning of software development life cycle
3. Since each phase has to be completed before we move to next phase issues will be identified and corrected in initial phase itself.
4. Since requirements are well understood and analyzed in beginning of project customer involvement in later phases is minimized.
5. High Visibility - The output of each stage gives more visibility on where we stand on progress of development.
6. This approach has control over deadline as work is distributed to teams in each stage.

Waterfall Model cons:

1. Not Flexible. Because this is a rigid model, Requirements cannot be changed throughout the cycle.
2. With rapid change in technology day by day we cannot have control to change the hardware and system requirements. Since system and hardware requirements are set up by customers in the beginning.
3. Small change in one phase leads to big change in each phase, as these phases are dependent on one another and eventually more time is consumed
4. If assumptions about implementations are wrong then estimated time for each phases exceeds and fails to meet deadline.
5. Resource idle time might increase as they have to be idle until previous stage is completely done. Therefore it is expensive.
6. Difficult to see the progress within phases.
7. Challenges, major issues, bottlenecks are identified in last stages in integration testing. So no

visibility of these in the beginning and high risk.

8. Product/Software deliverables are at the end of cycle.

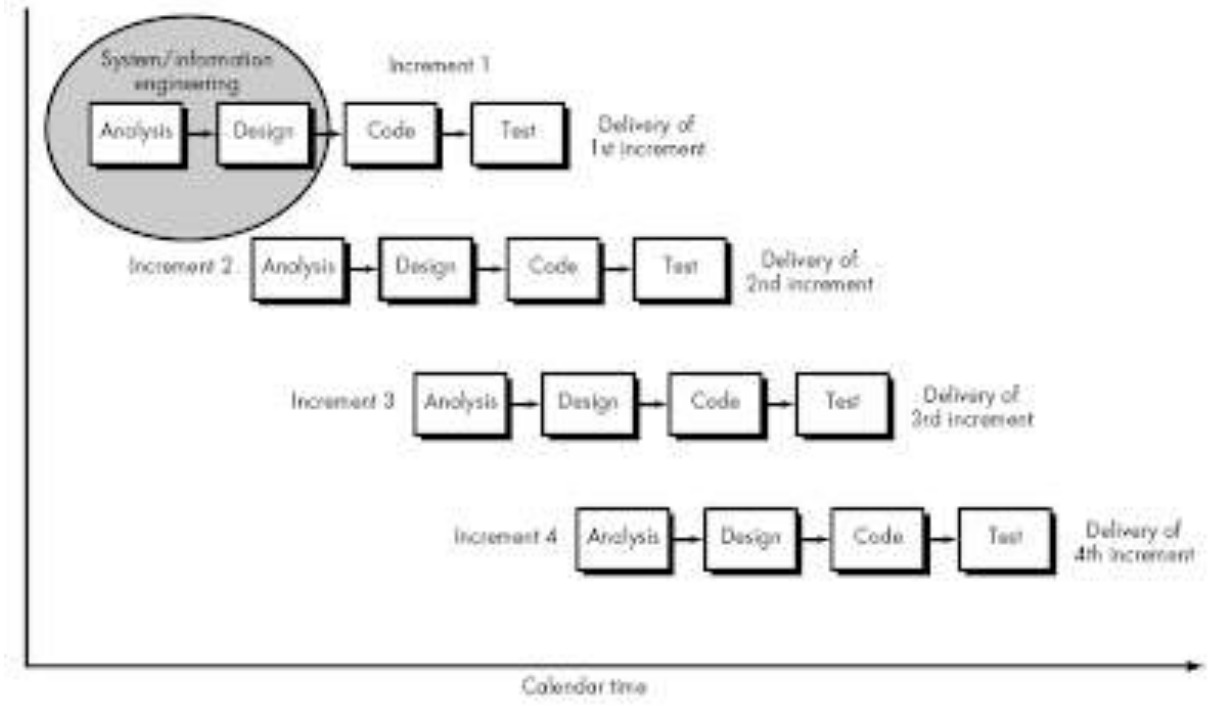
Q.5 Write short notes:

(a) Incremental model

(b) Prototype model

Ans (a) The incremental model combines elements of the linear sequential model (applied repetitively) with the iterative philosophy of prototyping. The incremental model applies linear sequences in a staggered fashion as calendar time progresses. Each linear sequence produces a deliverable "increment" of the software. For example, word-processing software developed using the incremental paradigm might deliver basic file management, editing, and document production functions in the first increment; more sophisticated editing and document production capabilities in the second increment; spelling and grammar checking in the third increment; and advanced page layout capability in the fourth increment. It should be noted that the process flow for any increment can incorporate the prototyping paradigm.

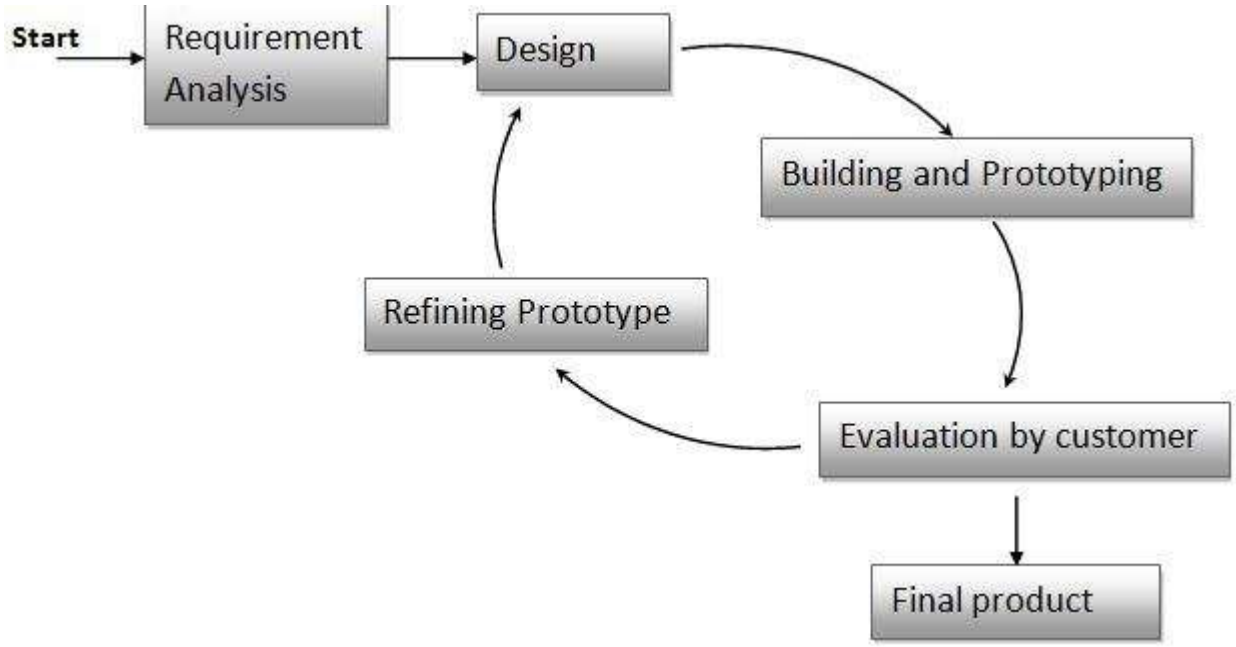
When an incremental model is used, the first increment is often a core product. That is, basic requirements are addressed, but many supplementary features (some known, others unknown) remain undelivered. The core product is used by the customer (or undergoes detailed review). As a result of use and/or evaluation, a plan is developed for the next increment. The plan addresses the modification of the core product to better meet the needs of the customer and the delivery of additional features and functionality. This process is repeated following the delivery of each increment, until the complete product is produced.



The incremental process model, like prototyping and other evolutionary approaches, is iterative in nature. But unlike prototyping, the incremental model focuses on the delivery of an operational product with each increment. Early increments are stripped down versions of the final product, but they do provide capability that serves the user and also provide a platform for evaluation by the user.

Incremental development is particularly useful when staffing is unavailable for a complete implementation by the business deadline that has been established for the project. Early increments can be implemented with fewer people. If the core product is well received, then additional staff (if required) can be added to implement the next increment. In addition, increments can be planned to manage technical risks. For example, a major system might require the availability of new hardware that is under development and whose delivery date is uncertain. It might be possible to plan early increments in a way that avoids the use of this hardware, thereby enabling partial functionality to be delivered to end-users without inordinate delay

Ans (b) Prototyping model is the model of software development life cycle where the Iterative process starts with a simple implementation of the software requirements and iteratively enhances the evolving versions until the full system is implemented.
Here is the diagrammatic view of the prototype model.



Few points to be noted:

1. Here, the developer and client interact to establish the requirements of the software.
2. The essence of prototyping is a quickly designed and can undergo immediate evaluation.
3. Here, the visible elements of the software, the input and the output are designed.
4. The final product of the design through this model is a prototype.
5. After the prototype is developed, the client evaluates the prototype and provides its recommendations and suggestion to the developer.
6. Until the all the user requirements are met, it continues in an iterative manner.

This question paper contains (1) no. of printed page.

JNIT JAGANNATH GUPTA INSTITUTE OF ENGINEERING & TECHNOLOGY JAIPUR

I-Mid Term Examination Session 2017-2018

B.Tech II Year IV Semester

Branch: CSE **Subject: POC**

Time: 10:00-11:30 **Subject Code: 4CS5A**

Date: 08 -03-18 **Max. Marks: 20**

Attempt any four questions out of following five questions

Q.1 Define Amplitude Modulation. Drive an expression for single tone AM Wave.

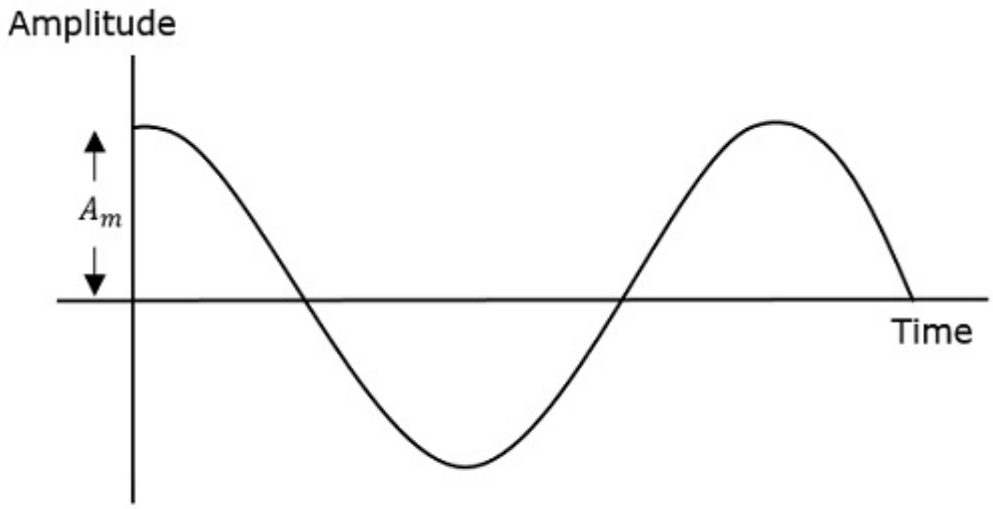
Ans. Among the types of modulation techniques, the main classification is Continuous-wave Modulation and Pulse Modulation. The continuous wave modulation techniques are further divided into Amplitude Modulation and Angle Modulation.

A continuous-wave goes on continuously without any intervals and it is the baseband message signal, which contains the information. This wave has to be modulated.

According to the standard definition, “The amplitude of the carrier signal varies in accordance with the instantaneous amplitude of the modulating signal.” Which means, the amplitude of the carrier signal which contains no information varies as per the amplitude of the signal, at each instant, which contains information. This can be well explained by the following figures.

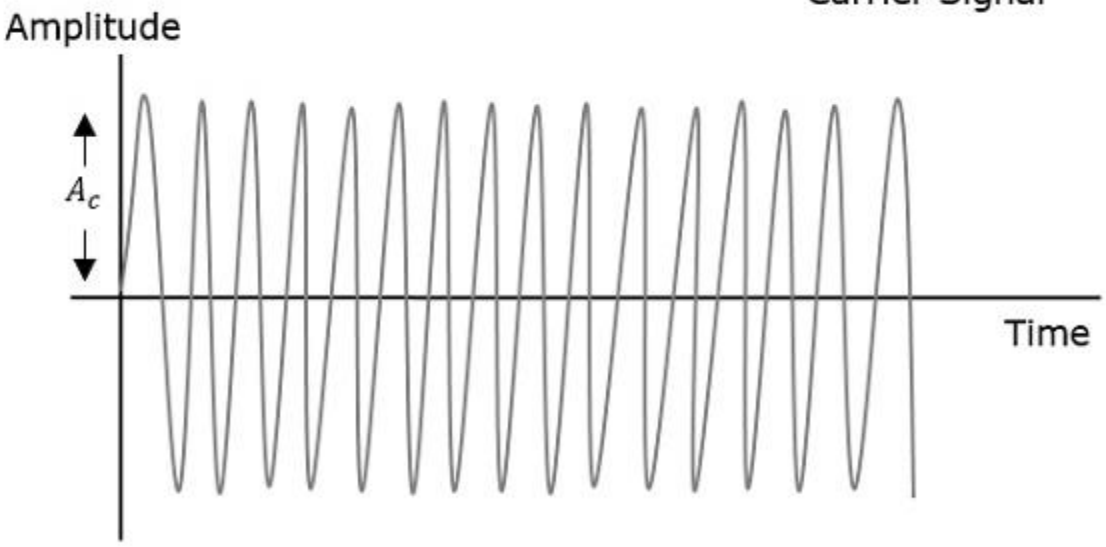
The modulating wave which is shown first is the message signal. The next one is the carrier wave, which is just a high frequency signal and contains no information. While the last one is the resultant modulated wave.

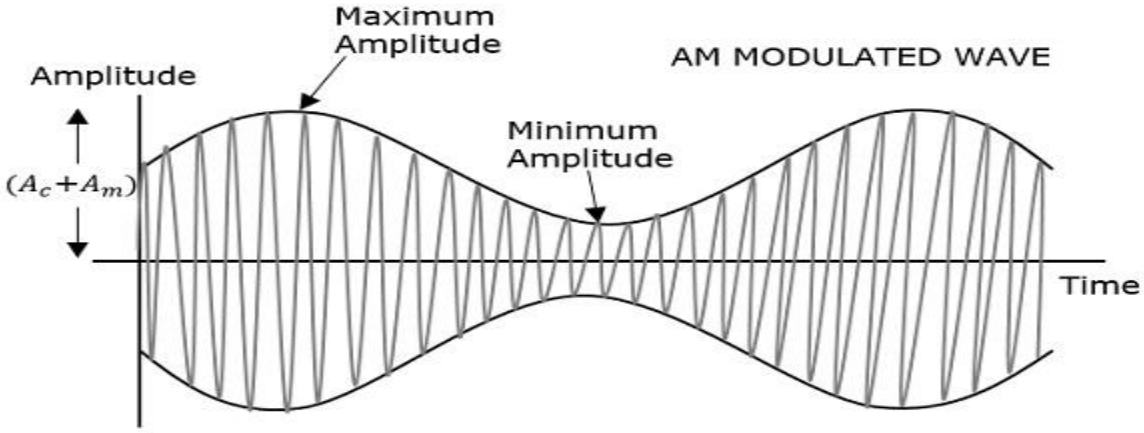
It can be observed that the positive and negative peaks of the carrier wave, are interconnected with an imaginary line. This line helps recreating the exact shape of the modulating signal. This imaginary line on the carrier wave is called as Envelope. It is the same as the message signal.



Base band signal

Carrier Signal





Mathematical Expression:

Following are the mathematical expression for these waves.

Time-domain Representation of the Waves

Let modulating signal be −

$$m(t)=Amcos(2\pi fmt)m(t)=Amcos(2\pi fmt)$$

Let carrier signal be −

$$c(t)=Accos(2\pi fct)c(t)=Accos(2\pi fct)$$

Where A_m = maximum amplitude of the modulating signal

A_c = maximum amplitude of the carrier signal

The standard form of an Amplitude Modulated wave is defined as −

$$S(t)=Ac[1+Kam(t)]cos(2\pi fct)S(t)=Ac[1+Kam(t)]cos(2\pi fct)$$

$$S(t)=Ac[1+\mu cos(2\pi fmt)]cos(2\pi fct)S(t)=Ac[1+\mu cos(2\pi fmt)]cos(2\pi fct)$$

$$Where,\mu=KaAmWhere,\mu=KaAm$$

Where A_m = maximum amplitude of the modulating signal

A_c = maximum amplitude of the carrier signal

The standard form of an Amplitude Modulated wave is defined as −

$$S(t)=Ac[1+Kam(t)]cos(2\pi fct)S(t)=Ac[1+Kam(t)]cos(2\pi fct)$$

$$S(t)=Ac[1+\mu cos(2\pi fmt)]cos(2\pi fct)S(t)=Ac[1+\mu cos(2\pi fmt)]cos(2\pi fct)$$

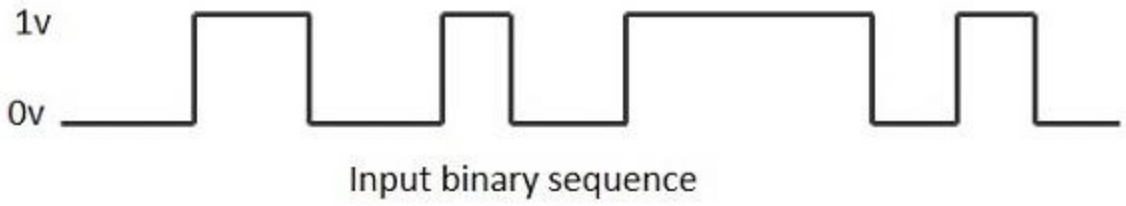
$$Where,\mu=KaAmWhere,\mu=KaAm$$

Q.2 Explain ON-OFF keying with its generation and demodulation?

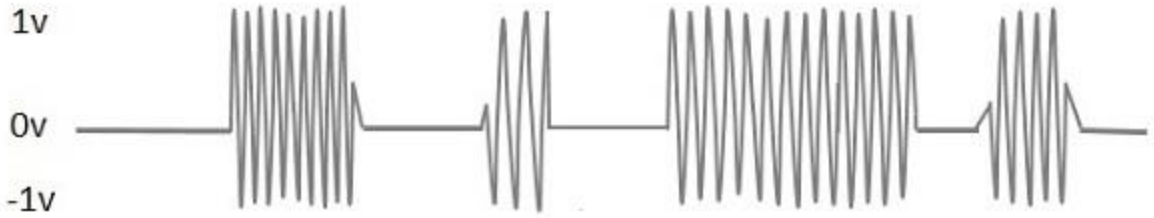
ANS Amplitude Shift Keying (ASK) is a type of Amplitude Modulation which represents the binary data in the form of variations in the amplitude of a signal.

Any modulated signal has a high frequency carrier. The binary signal when ASK modulated, gives a zero value for Low input while it gives the carrier output for High input.

The following figure represents ASK modulated waveform along with its input.



Input binary sequence



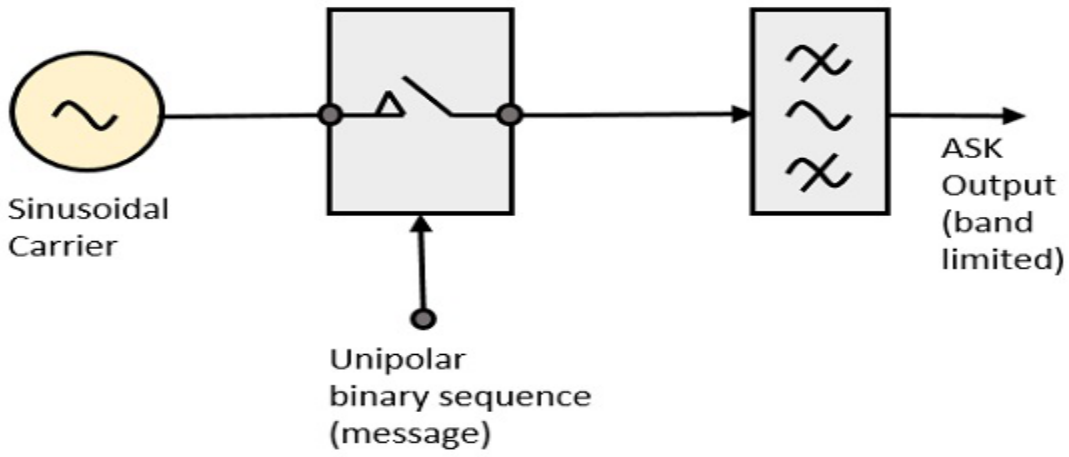
ASK Modulated output wave

To find the process of obtaining this ASK modulated wave, let us learn about the working of the ASK modulator.

ASK Modulator

The ASK modulator block diagram comprises of the carrier signal generator, the binary sequence from the message signal and the band-limited filter. Following is the block diagram of the ASK Modulator.

ASK Generation method



The carrier generator, sends a continuous high-frequency carrier. The binary sequence from the message signal makes the unipolar input to be either High or Low. The high signal closes the switch, allowing a carrier wave. Hence, the output will be the carrier signal at high input. When

there is low input, the switch opens, allowing no voltage to appear. Hence, the output will be low.

The band-limiting filter, shapes the pulse depending upon the amplitude and phase characteristics of the band-limiting filter or the pulse-shaping filter.

ASK Demodulator

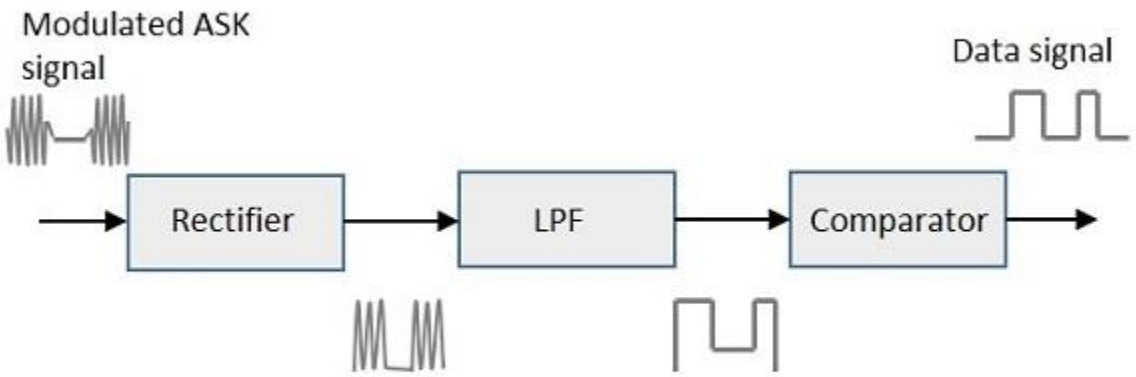
There are two types of ASK Demodulation techniques. They are −

- Asynchronous ASK Demodulation/detection
- Synchronous ASK Demodulation/detection

The clock frequency at the transmitter when matches with the clock frequency at the receiver, it is known as a Synchronous method, as the frequency gets synchronized. Otherwise, it is known as Asynchronous.

Asynchronous ASK Demodulator

The Asynchronous ASK detector consists of a half-wave rectifier, a low pass filter, and a comparator. Following is the block diagram for the same.

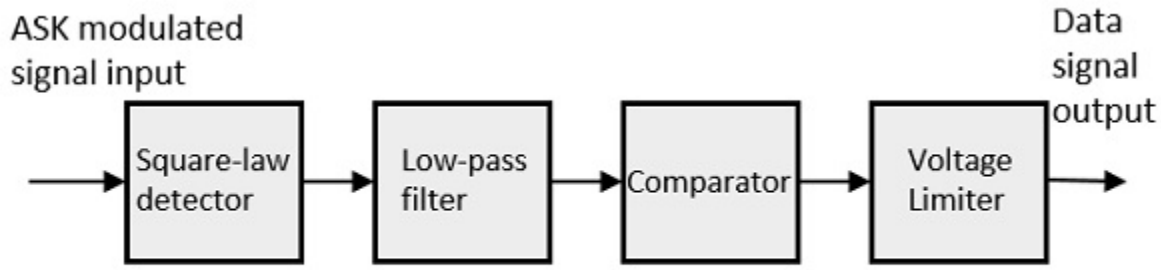


Asynchronous ASK detector

The modulated ASK signal is given to the half-wave rectifier, which delivers a positive half output. The low pass filter suppresses the higher frequencies and gives an envelope detected output from which the comparator delivers a digital output.

Synchronous ASK Demodulator

Synchronous ASK detector consists of a Square law detector, low pass filter, a comparator, and a voltage limiter. Following is the block diagram for the same.



Synchronous ASK detector

The ASK modulated input signal is given to the Square law detector. A square law detector is one whose output voltage is proportional to the square of the amplitude modulated input voltage. The low pass filter minimizes the higher frequencies. The comparator and the voltage limiter help to get a clean digital output.

Q.3 what is Base-Band Transmission. Design following data in PAM Format and Explain Mathematically

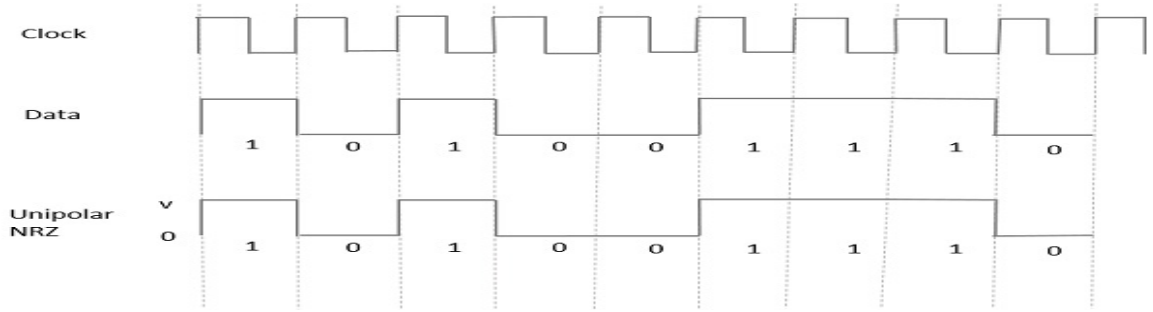
Data: 1 0 1 0 0 1 1 1 0

Ans.

Unipolar Non-Return to Zero (NRZ)

In this type of unipolar signaling, a High in data is represented by a positive pulse called as Mark, which has a duration T_0 equal to the symbol bit duration. A Low in data input has no pulse.

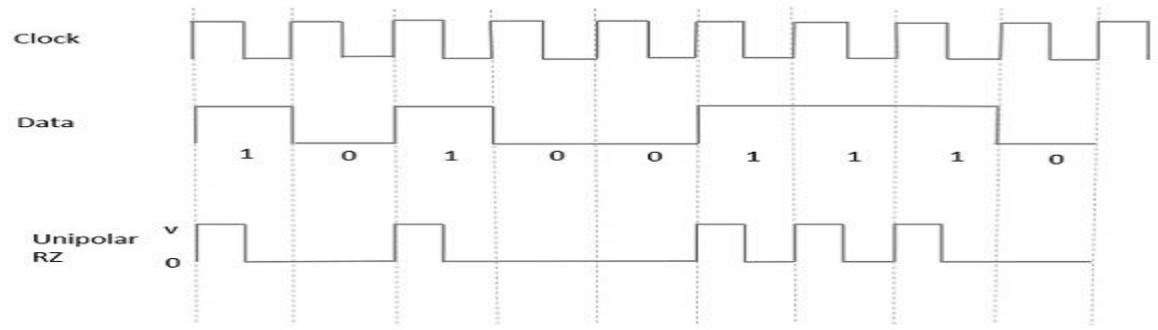
The following figure clearly depicts this.



Unipolar Return to Zero (RZ)

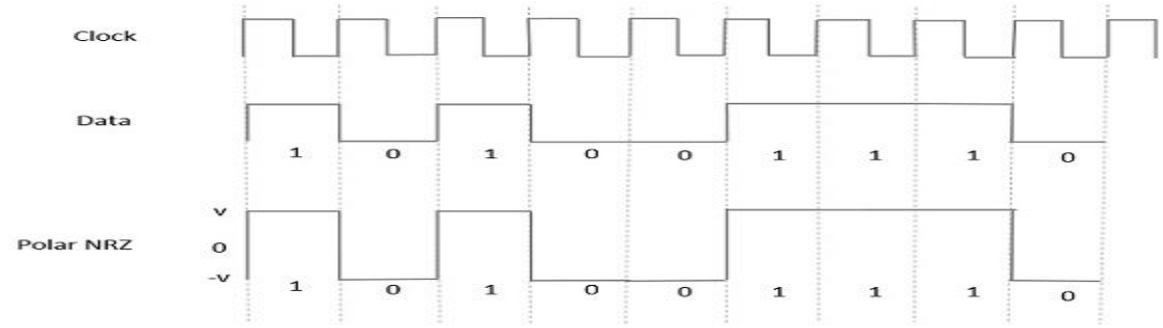
In this type of unipolar signaling, a High in data, though represented by a Mark pulse, its duration T_0 is less than the symbol bit duration. Half of the bit duration remains high but it immediately returns to zero and shows the absence of pulse during the remaining half of the bit duration.

It is clearly understood with the help of the following figure.



Polar NRZ

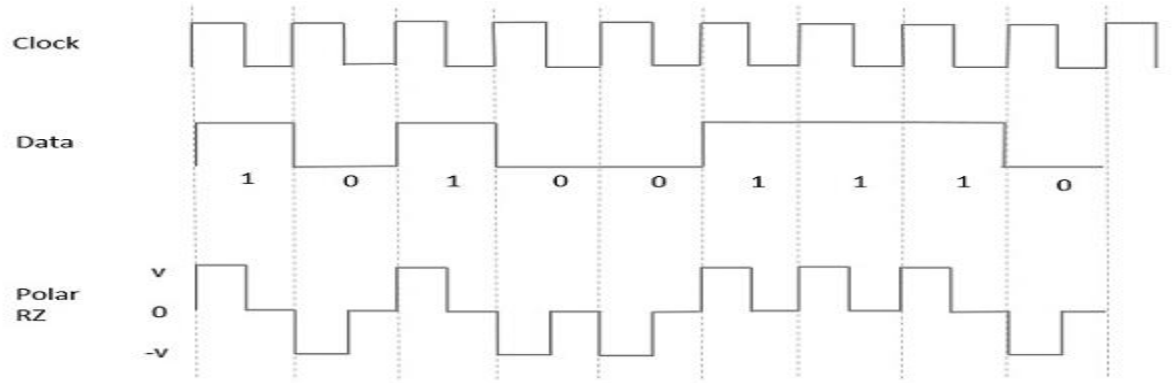
In this type of Polar signaling, a High in data is represented by a positive pulse, while a Low in data is represented by a negative pulse. The following figure depicts this well.



Polar RZ

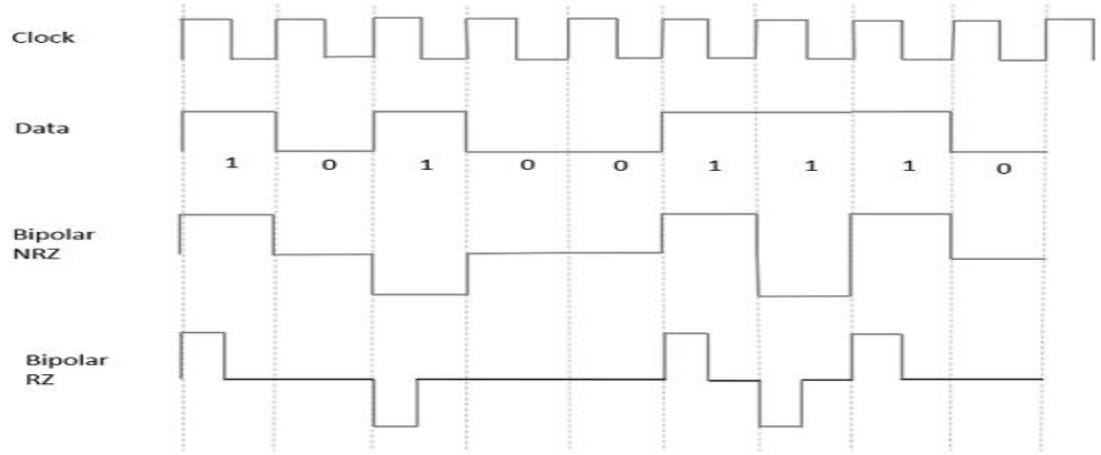
In this type of Polar signaling, a High in data, though represented by a Mark pulse, its duration T_0 is less than the symbol bit duration. Half of the bit duration remains high but it immediately returns to zero and shows the absence of pulse during the remaining half of the bit duration.

However, for a Low input, a negative pulse represents the data, and the zero level remains same for the other half of the bit duration. The following figure depicts this clearly.



AMI Code

An example of this type is Alternate Mark Inversion (AMI). For a 1, the voltage level gets a transition from + to – or from – to +, having alternate 1sto be of equal polarity. A 0 will have a zero voltage level.



Q.4 Explain BPSK technique in details. Make suitable diagram.

Ans. Phase Shift Keying (PSK) is the digital modulation technique in which the phase of the carrier signal is changed by varying the sine and cosine inputs at a particular time. PSK technique is widely used for wireless LANs, bio-metric, contactless operations, along with RFID and Bluetooth communications.

PSK is of two types, depending upon the phases the signal gets shifted. They are −

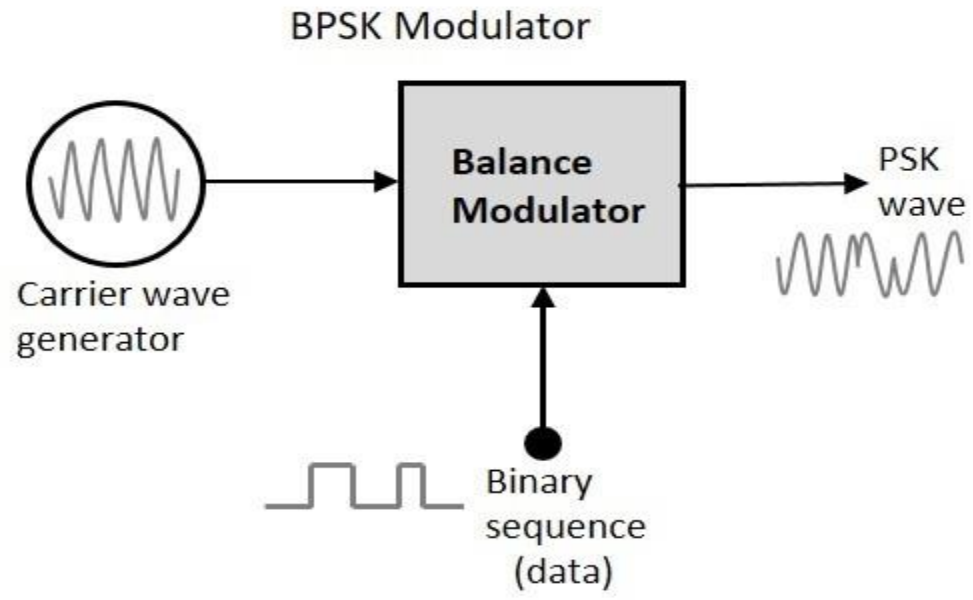
Binary Phase Shift Keying (BPSK)

This is also called as 2-phase PSK or Phase Reversal Keying. In this technique, the sine wave carrier takes two phase reversals such as 0° and 180°.

BPSK is basically a Double Side Band Suppressed Carrier (DSBSC) modulation scheme, for message being the digital information.

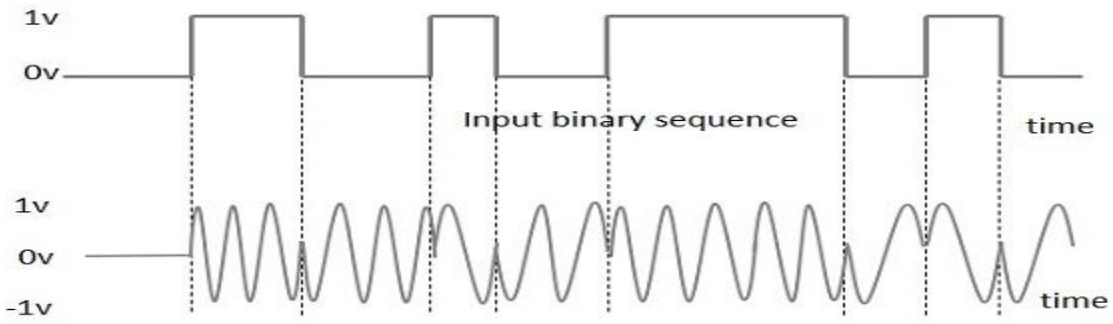
BPSK Modulator

The block diagram of Binary Phase Shift Keying consists of the balance modulator which has the carrier sine wave as one input and the binary sequence as the other input. Following is the diagrammatic representation.



The modulation of BPSK is done using a balance modulator, which multiplies the two signals applied at the input. For a zero binary input, the phase will be 0° and for a high input, the phase reversal is of 180°.

Following is the diagrammatic representation of BPSK Modulated output wave along with its given input.

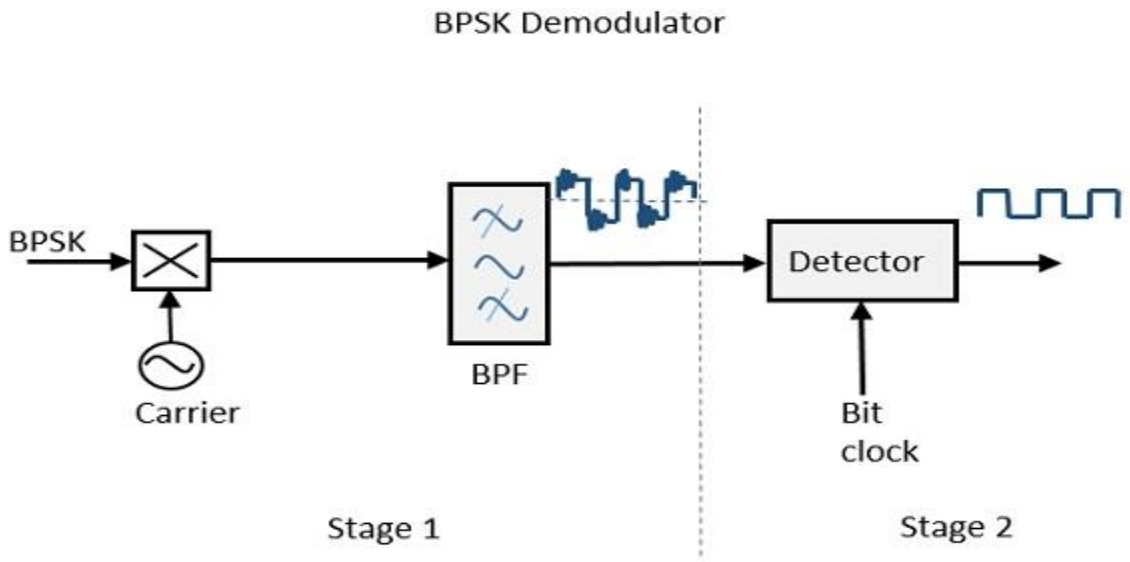


BPSK Modulated output wave

The output sine wave of the modulator will be the direct input carrier or the inverted (180° phase shifted) input carrier, which is a function of the data si

BPSK Demodulator

The block diagram of BPSK demodulator consists of a mixer with local oscillator circuit, a bandpass filter, a two-input detector circuit. The diagram is as follows.



By recovering the band-limited message signal, with the help of the mixer circuit and the band pass filter, the first stage of demodulation gets completed. The base band signal which is band limited is obtained and this signal is used to regenerate the binary message bit stream.

In the next stage of demodulation, the bit clock rate is needed at the detector circuit to produce the original binary message signal. If the bit rate is a sub-multiple of the carrier frequency, then the bit clock regeneration is simplified. To make the circuit easily understandable, a decision-making circuit may also be inserted at the 2^{nd} stage of detection.

Q.5 writes Short Notes on Following (any two)

(a) Difference between AM and FM.

AM V/S FM		
Stands for	Amplitude Modulation	Frequency Modulation
Origin	AM method of audio transmission was first successfully carried out in the mid 1870s.	FM radio was developed in the United states in the 1930s, mainly by Edwin Armstrong.
Modulating differences	In AM, a radio wave known as the "carrier" or "carrier wave" is modulated in amplitude by the signal that is to be transmitted. The frequency and phase remain the same.	In FM, a radio wave known as the "carrier" or "carrier wave" is modulated in frequency by the signal that is to be transmitted. The amplitude and phase remain the same.
Pros and cons	AM has poorer sound quality compared with FM, but is cheaper and can be transmitted over long distances. It has a lower bandwidth so it can have more stations available in any frequency range.	FM is less prone to interference than AM. However, FM signals are impacted by physical barriers. FM has better sound quality due to higher bandwidth.
Frequency Range	AM radio ranges from 535 to 1705 KHz (OR) Up to 1200 bits per second.	FM radio ranges in a higher spectrum from 88 to 108 MHz. (OR) 1200 to 2400 bits per second.
Bandwidth Requirements	Twice the highest modulating frequency. In AM radio broadcasting, the modulating signal has bandwidth of 15kHz, and hence the bandwidth of an amplitude-modulated signal is 30kHz.	Twice the sum of the modulating signal frequency and the frequency deviation. If the frequency deviation is 75kHz and the modulating signal frequency is 15kHz, the bandwidth required is 180kHz.
Zero crossing in modulated signal	Equidistant	Not equidistant
Complexity	Transmitter and receiver are simple but syncronization is needed in case of SSBSC AM carrier.	Tranmitter and reciver are more complex as variation of modulating signal has to beconverted and detected from corresponding variation in frequencies.(i.e. voltage to frequency and frequency to voltage conversion has to be done).
Noise	AM is more susceptible to noise because noise affects amplitude, which is where	FM is less susceptible to noise because information in an FM signal is

AM V/S FM	
information is "stored" in an AM signal.	transmitted through varying the frequency, and not the amplitude.

(b) AMI Code & Split phase Manchester code

ANS. **Bipolar Signaling**

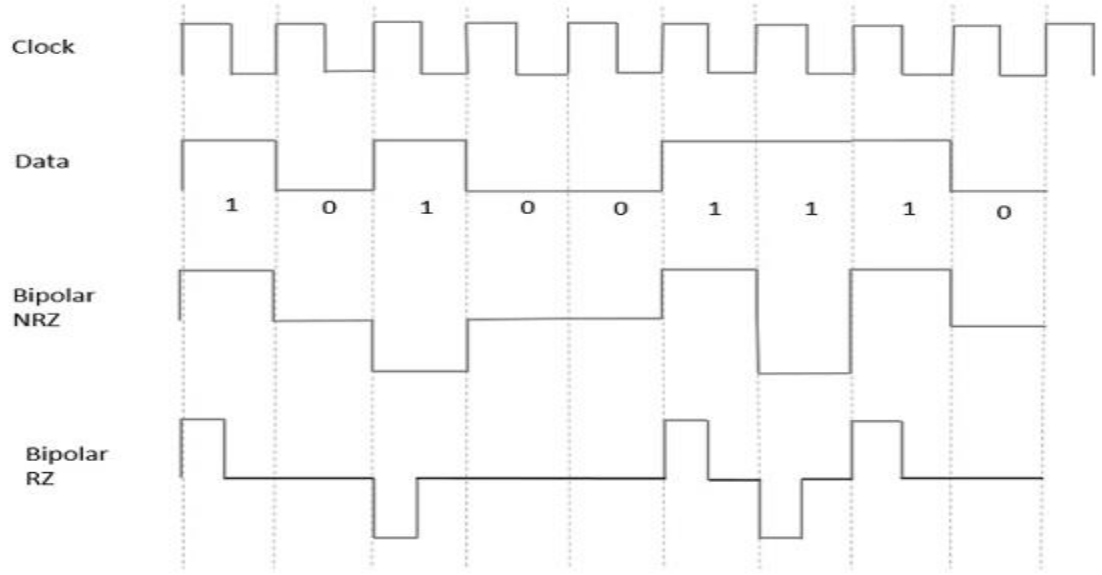
This is an encoding technique which has three voltage levels namely +, - and 0. Such a signal is called as duo-binary signal.

An example of this type is Alternate Mark Inversion (AMI). For a 1, the voltage level gets a transition from + to – or from – to +, having alternate 1sto be of equal polarity. A 0 will have a zero voltage level.

Even in this method, we have two types.

- Bipolar NRZ
- Bipolar RZ

From the models so far discussed, we have learnt the difference between NRZ and RZ. It just goes in the same way here too. The following figure clearly depicts this.



(c) Inter symbol Interference (ISI)

ANS. Inter Symbol Interference

This is a form of distortion of a signal, in which one or more symbols interfere with subsequent signals, causing noise or delivering a poor output.

Causes of ISI

The main causes of ISI are −

- Multi-path Propagation
- Non-linear frequency in channels

The ISI is unwanted and should be completely eliminated to get a clean output. The causes of ISI should also be resolved in order to lessen its effect.

To view ISI in a mathematical form present in the receiver output, we can consider the receiver output.

The receiving filter output y(t) is sampled at time ti=iTb (with i taking on integer values), yielding −

$$y(ti)=\mu\infty\sum k=-\infty akp(iTb-kTb) =\mu ai+\mu\infty\sum k=-\infty k\neq\bullet\ iakp(iTb-kTb)$$

In the above equation, the first term μai is produced by the i^{th} transmitted bit.

The second term represents the residual effect of all other transmitted bits on the decoding of the i^{th} bit. This residual effect is called as Inter Symbol Interference.

In the absence of ISI, the output will be −

$$y(ti)=\mu ai$$

This equation shows that the i^{th} bit transmitted is correctly reproduced. However, the presence of ISI introduces bit errors and distortions in the output.

While designing the transmitter or a receiver, it is important that you minimize the effects of ISI, so as to receive the output with the least possible error rate.

This question paper contains (1) no. of printed page.

JNIT JAGANNATH GUPTA INSTITUTE OF ENGINEERING & TECHNOLOGY JAIPUR

I-Mid Term Examination Session 2018

B.Tech II Year IV Semester

Branch: CS **Subject: PPL**

Time: **Subject Code: 4CS 6**

Date: **Max. Marks: 20**

Attempt any four questions out of following five questions

Q1. What is Programming Language? Explain Attributes of Good Programming language.

A programming language is a formal language that specifies a set of instructions that can be used to produce various kinds of output. Programming languages generally consist of instructions for a computer. Programming languages can be used to create programs that implement specific algorithms.

Several characteristics believed to be important for making a programming language good are:

Simplicity: A good programming language must be simple and easy to learn and use. It should provide a programmer with a clear, simple and unified set of concepts, which can be easily grasped. The overall simplicity of a programming language strongly affects the readability of the programs written in that language, and programs, which are easier to read and understand, are also easier to maintain. It is also easy to develop and implement a compiler or an interpreter for a programming language, which is simple. However, the power needed for the language should not be sacrificed for simplicity.

Naturalness:- A good language should be natural for the application area, for which it has been designed. That is, it should provide appropriate operators, data structures, control structures, and a natural syntax to facilitate the users to code their problem easily and efficiently.

Abstraction:- Abstraction means the ability to define and then use complicated structures or operations in ways that allow many of the details to be ignored. The degree of abstraction allowed by a programming language directly effects its writ ability. Object oriented language support high degree of abstraction. Hence, writing programs in object oriented language is much easier. Object oriented language also support re usability of program segments due to this features.

Efficiency :- Programs written in a good programming language are efficiently translated into machine code, are efficiently executed, and acquire as little space in the memory as possible. That is a good programming language is supported with a good language translator which gives due consideration to space and time efficiency.

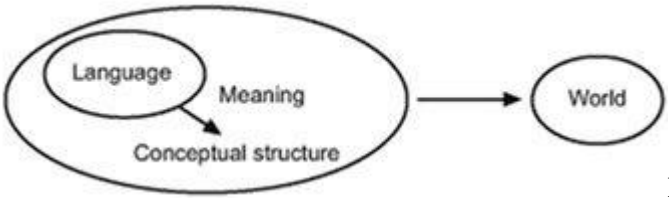
Structured:- Structured means that the language should have necessary features to allow its users to write their programs based on the concepts of structured programming. This property of a moreover, it forces a programmer to look at a problem in a logical way, so that fewer errors are created while writing a program for the problem.

Compactness :- In a good programming language, programmers should be able to express intended operations concisely. A verbose language is generally not liked by programmers, because they need to write too much.

Locality :- A good programming language should be such that while writing a programmer concentrate almost solely on the part of the program around the statement currently being worked with.

Q2. What is Syntax and Semantics? Explain General Syntactic criteria of a Programming Language.

ANS: Semantics and Syntax are two different fields of micros linguistics. Semantics deals with the study of words without any consideration given to their meanings. On the other hand, Syntax is the study which deals with analyzing that how words are combined in order to form grammatical sentences.



Linguistics is the study of language. Broadly, linguistics is divided into branches of General linguistic, micro linguistic and macro linguistic. Syntax and Semantics fall into the category of micro linguistic which is concerned with the internal view of language.

Semantics deals with the study of words with respect to their meaning irrespective of the context. On the other hand, Syntax is the study which deals with analyzing that how words are combined in order to form grammatical sentences. In simple language, syntax is all about what the grammar allows, whereas semantics is what it means.

Theoretically, Semantics primarily consider the meanings of entities like words and sentences. However, it does not account for other associated entities like text/discourse, paragraph, phrase or morpheme. Semantics can also be described as the study of the relation between form and meaning.

Syntax has been derived from the Greek words which mean ‘together’ or ‘arrangement’ and ordering. Syntax deals with the study of formation of sentences describing that how words combine together to form bigger units than words like phrases or sentences. These phrases or sentences are basically proper structured strings.

Structure is very important in making out some sort of sense from a statement. A small structural mistake may make any sentence senseless. It also directly refers to the rules and principles that guide the sentences formation (structure) of any individual language.

In order to understand the structure of the sentence one must be familiar with phrases, modifiers, noun phrase, etc. Syntax tress help in explaining sentence structures.

Thus, semantics and syntax focus on two different issues related to linguistics. Semantics is all about meanings of words and sentences, whereas syntax is about the formation of sentences. In the case of semantics, a sentence in which words are not ordered properly can be interpreted by few people on the basis of their prior knowledge. However, the same sentence is meaningless in terms of syntax, as syntax only deals with linguistically and grammatically correct sentences.

Comparison between Semantics and Syntax:

	Semantics	Syntax
Definition	Semantics is a term which is derived from the Greek word seme meaning sigh. Semantics is another important field related to theoretical linguistics. It is all about studying the meaning of linguistic expressions.	Syntax is the study which deals with analyzing that how words are combined in order to form grammatical sentences.
Related to	Meanings of words and sentences	The structure of words
Rules	Describe the relationship between symbol and the things they mean or refer to	Describe the correct word order and inflectional structure in sentences
Main aspect	Relation between form and meaning	Word order
Approach towards meaning of a sentence	Individual's own interpretation on the basis of previous knowledge	Linguistically and grammatically correct

The primary purpose of syntax is to provide a notation for communication between the programmer and the programming language processor. The choice of particular syntactic structures, however, is constrained only slightly by the necessity to communication particular items of information. The details of syntax are chosen largely on the basis of secondary criteria, such as readability, which are unrelated to the primary goal of communicating information to the language processor. There are many secondary criteria, but they may be roughly categorized under the general goals of making programs easy to read, easy to write, easy to translate and unambiguous. These are –

- Readability – A program is readable if the underlying structure of the algorithm and data represented by the program is apparent from an inspection of the program text. A readable program is often said to be self-documenting. That is, it is understandable without any separate documentation. Readability, of course, cannot be guaranteed by the design of a language, because even the best design may be circumvented by poor programming. Readability is enhanced by a program syntax in which syntactic differences reflect underlying semantic differences so that program constructs that do similar things look similar and program constructs that program constructs that do radically different things look different. In general the greater the variety of syntactic constructs used, the more easily the program structure may be made to reflect different underlying semantic structures. Languages that provide only a few different syntactic constructs in general lead to less readable programs.
- Writeability – The syntactic features that make a program easy to write are often in conflict with those features that make it easy read. Writeability is enhanced by use of concise and regular syntactic structures, whereas for readability a variety of more verbose constructs are helpful. Implicit syntactic conventions that allow declarations and operations to be left unspecified make programs shorter and easier to write but harder to read. A syntax is redundant if it communicates the same item of information in more than one way. Some redundancy is useful in programming language syntax because it makes a program easier to read and also allows for error checking during translation. The disadvantage is that redundancy makes program more verbose and thus harder to write.
- Ease of Verifiability – related to readability and writeability is the concept of program correctness or program verification. After many years of experience, we now understand that understanding each programming language statement is relatively easy, but the overall process of creating correct programs is extremely difficult. Therefore, we need techniques that enable the program to be mathematically proved correct.
- Lack of Ambiguity – Ambiguity is a central problem in every language design. An ambiguous construction allows two or more different interpretations. The problems of ambiguity usually

arise not in the structure of individual program elements but in the interplay between different structures.

Q3. What is structured data type? Explain Specification and Implementation of structured data type.

ANS: A data structure is a data object that contains other data objects as its elements or components.

1. **Specifications**
   - Number of components

     Fixed size - Arrays
     Variable size – stacks, lists. Pointer is used to link components.

   - Type of each component

     Homogeneous – all components are the same type
     Heterogeneous – components are of different types

   - Selection mechanism to identify components – index, pointer

     Two-step process:
       referencing the structure
       selection of a particular component

   - Maximum number of components
   - Organization of the components: simple linear sequence
     - simple linear sequence
     - multidimensional structures:
       - separate types (Fortran)
       - vector of vectors (C++)

   Operations on data structures

   - Component selection operations

     Sequential
     Random

   - Insertion/deletion of components
   - Whole-data structure operations
     Creation/destruction of data structures

2. **Implementation of data structure types**

   Storage representation

Includes:

a. storage for the components
b. optional descriptor - to contain some or all of the attributes

Sequential representation: the data structure is stored in a single contiguous block of storage, that includes both descriptor and components. Used for fixed-size structures, homogeneous structures (arrays, character strings)

Linked representation: the data structure is stored in several noncontiguous blocks of storage, linked together through pointers. Used for variable-size structured (trees, lists)

Stacks, queues, lists can be represented in either way. Linked representation is more flexible and ensures true variable size, however it has to be software simulated.

Implementation of operations on data structures

Component selection in sequential representation: Base address plus offset calculation. Add component size to current location to move to next component.

Component selection in linked representation: Move from address location to address location following the chain of pointers.

Storage management

Access paths to a structured data object - to endure access to the object for its processing. Created using a name or a pointer.

Two central problems:

Garbage – the data object is bound but access path is destroyed.
Memory cannot be unbound.

Dangling references – the data object is destroyed, but the access path still exists.

3. **Declarations and type checking for data structures**

What is to be checked:

- Existence of a selected component
- Type of a selected component

4. **Vectors and arrays**

A vector - one dimensional array

A matrix - two dimensional array

Multidimensional arrays

A slice - a substructure in an array that is also an array, e.g. a column in a matrix.

Implementation of array operations:

. **Access -** can be implemented efficiently if the length of the components of the array is known at compilation time. The address of each selected element can be computed using an arithmetic expression.

a. **Whole array operations,** e.g. copying an array - may require much memory.

Associative arrays

Instead of using an integer index, elements are selected by a key value, that is a part of the element. Usually the elements are sorted by the key and binary search is performed to find an element in the array.

5. **Records**

A record is a data structure composed of a fixed number of components of different types.
The components may be heterogeneous, and they are named with symbolic names.

Specification of attributes of a record:

Number of components
Data type of each component
Selector used to name each component.

Implementation:

Storage: single sequential block of memory where the components are stored sequentially.

Selection: provided the type of each component is known, the location can be computed at translation time.

Note on efficiency of storage representation:

For some data types storage must begin on specific memory boundaries (required by the hardware organization). For example, integers must be allocated at word boundaries (e.g. addresses that are multiples of 4). When the structure of a record is designed, this fact has to be taken into consideration. Otherwise the actual memory needed might be more than the sum of the length of each component in the record. Here is an example:

```
struct employee
  { char Division;
    int  IdNumber; };
```

The first variable occupies one byte only. The next three bytes will remain unused and then the second variable will be allocated to a word boundary.
Careless design may result in doubling the memory requirements.

Q4. **Explain Type Checking and Type conversion with Example.**

Type conversion

- Type conversion occurs when the expression has data of mixed data types.
- example of such expression include converting an integer value in to a float value, or assigning the value of the expression to a variable with different data type.
- In type conversion, the data type is promoted from lower to higher because converting higher to lower involves loss of precision and value.
- For type conversion, C following some General rules explained below
    - Integer types are lower than floating point types
    - Signed types are lower than unsigned types
    - Short whole number types are lower than longer types
    - **double>float>long>int>short>char**

Type Conversion refers to translating of values (roughly speaking, contents of the object), so that they may be interpreted as belonging to a new type.

Example :

```
double  d;
long    l;
int     i;

if (d > i)   d = i;
if (i > l)   l = i;
if (d == l)  d *= 2;
```

Type Casting (or) Explicit Type conversion:

- Explicit type conversions can be forced in any expression , with a unary operator called a cast.
- **Syntax is**

    (type-name) expression;

- **Example**

```
int n;
float x;
x=(float)n;
```

- The above statement will convert the value of n to a float value before assigning to x.but n is not altered
- Type casting does not change the actual value of the variable but the resultant value may be put in temporary storage.
- The cast operator has the same high precedence as other unary operators.
- The Typecasting should not be used in some places.
- Type cast should not be used to override a const or volatile declaration. Overriding these type modifiers can cause the program to fail to run correctly.

- Type cast should not be used to turn a pointer to one type of structrure or data type in to another.
- **Example** of type casting using pointers

```
#include<stdio.h>
main()
{
   void *temp; //void pointer
   char c='a',*ch="hello";
   int i=10;
   temp=&c;
   printf("char=%c\n",*(char *)temp);
   temp=ch;
   printf("string=%s\n",(char *)temp);
   temp=&i;
   printf("i=%d\n",*(int *)temp);
   return 0;
}
```

Q5. Explain Exception Handling in detail.

An Exception is an unwanted event that interrupts the normal flow of the program. When an exception occurs program execution gets terminated. In such cases we get a system generated error message. The good thing about exceptions is that they can be handled in Java. By handling the exceptions we can provide a meaningful message to the user about the issue rather than a system generated message, which may not be understandable to a user.

Why an exception occurs?

There can be several reasons that can cause a program to throw exception. For example: Opening a non-existing file in your program, Network connection problem, bad input data provided by user etc.

Exception Handling

If an exception occurs, which has not been handled by programmer then program execution gets terminated and a system generated error message is shown to the user.

Difference between error and exception

Errors indicate that something severe enough has gone wrong, the application should crash rather than try to handle the error.

Exceptions are events that occur in the code. A programmer can handle such conditions and take necessary corrective actions. Few examples:
Null Pointer Exception – When you try to use a reference that points to null.
Arithmetic Exception – When bad data is provided by user, for example, when you try to divide a number by zero this exception occurs because dividing a number by zero is undefined.

Three statements play a part in handling exceptions:

- The *try* statement identifies a block of statements within which an exception might be thrown.
- The *catch* statement must be associated with a try statement and identifies a block of statements that can handle a particular type of exception. The statements are executed if an exception of a particular type occurs within the try block.
- The Throw statement must be associated with a try statement and identifies a block of statements that are executed regardless of whether or not an error occurs within the try block.

Here's the general form of these statements:

```
try {
   statement(s)
}
catch (exceptiontype name)
 {
   statement(s)
}
throw {
   statement(s)
}
```

s