07/07/25

1. Write a C program and create array and ~~creat~~ which stores 5 subs and calculates %.

```
#include <stdio.h>

int main () {
 int marks [5];
 float percentage;
 int i;

printf ("Enter marks (out of 40) for 5 subjects :\n");

for (i=0; i<5; i++) {
 printf ( "subject %d : ", i + 1);
 scanf ( "%d", &marks [i]);
 total += marks[i]);
 }

percentage = (total / 200.0) * 100;

printf ("\n Total Marks= %d out of 200\n", total);
printf (" percentage = %.2f %% \n", percentage);

return 0;

}
```

Output :

Enter marks (out of 40) for 5 subjects :

Subject 1 : 34
Subject 2 : 38
Subject 3 : 31
Subject 4 : 39
Subject 5 : 37

Total Marks = 179 out of 200
Percentage = 89.50%

2) #include <stdio.h>

```
int main () {
  int num;

printf ("Enter a number: ");
scanf ("%d", & num);

if (num %2 == 0) {
   printf ("%d is even.\n", num);
  }

else

{
  printf("%d is odd.\n",num);
 }

return 0;
}
```

Enter a number : 32

32 is even

3 #include <stdio.h>

```
int main () {
   int num, i, isprime;

printf ("Prime numbers between 1&50 are: \n");

for (num=2; num<=50; num++) {
 is Prime = 1;

for (i=2; i<num ; i++) {
  if (num %. i ==0) {
     is prime =0;
     break;
  }
}

if (is prime == 1) {
      printf ("%d", num);
   }
 }

printf ("\n");

return 0;
}
```

Prime numbers between 1 & 50 are:

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47

5

```
#include <stdio.h>

int main () {
   int num, original, reversed = 0, digit;

printf ("Enter a number : ");
scanf ("%d", &num);

original = num;

while (num != 0) {
     digit = num % 10;
    reversed = reversed * 10 + digit;
     num = num / 10;
  }

if (original == reversed) {
   printf ("%d is a palindrome.\n", original);
  }
else
{
 printf ("%d is not a palindrome.\n", original);
}

return 0;
 }
```

Enter a number : 51
51 is not a ~~number~~ palindrome

Enter a number : 66
66 is a palindrome

★ Experiment - I

Q1) Find an average of 5 nos using array.

Q2) Display :

```
*
# #
* * *
# # # #
```

Q3) Find first repeating numbers in array

Q4) Find greatest & smallest element in the array

Q5) Square the odd position in array

1. # include <stdio.h>

```
int main () {
  int arr[5]
  int sum = 0;
  float average;

printf ("Enter numbers : \n ");
 for (int =0 ; i<5 i++) {
   scanf ("%d ,&arr[i]);
  sum+ = arr[i];
 }

average =sum /5.0;

printf ("Average = %.2f \n", average);

return 0;
 }
```

Enter numbers:
10 20 30 40 50

Average: 30.00

2)

```
#include <stdio.h>

int main () {
    int rows = 4;

 for (int = 1, i<rows; i++) {
    char symbol = (i%2 == 0) ? '#' : '*';

 for (int j=1 ; j <= i ; j++) {
 printf ("%c", symbol);
  }

 printf ("\n");
  }
return 0;
}
```

```
*
# #
* * *
# # # #
```

3) #include <stdio.h>

```
int main () {

int arr [] = {4,5, 1,2,3,5,1};
int n = sizeof (arr) / sizeof arr [0] );
int found = 0;

for (int =0; i<n-1; i++) {
   for (int j=i+1; j<n; j++){
   if(arr [i] = = arr[ j ]) {
    printf ("First number is : %d \n", arr[i]);
    found= 1;
    break;
    }
 }
  if (found) break ;
}
if (! found ) {
  printf("No repeating elements found. \n");
  }
 return 0;
}
```

int arr[] = {4,5,1,2,3,5,1}

first repeating number = 5

Q4)

```
#include <stdio.h>

int main() {
    int n;

printf("Enter number of elements : ");
scanf("%d", &n);

int arr[n];

printf("Enter %d numbers :\n", n);
for (int i=0; i<n; i++) {
    scanf("%d", &arr[i]);
}

int smallest = arr[0];
int greatest = arr[0];

for (int i = 1; i<n; i++) {
    if (arr[i] < smallest)
        smallest = arr[i];
    if (arr[i] > greatest)
        greatest = arr[i];
}

printf("Smallest element : %d\n", smallest);
printf("Greatest element : %d\n", greatest);

return 0;
}
```

Enter number of element : 5

Enter 5 numbers:

1, 2, 3, 4, 5

smallest element : 1
Greatest element : 5

5)

```
#include <stdio.h>

int main () {

int n;

printf ("Enter number of elements: ");
scanf ("%d", &n);

int arr[n]

printf ("Enter %d numbers: \n", n);
for (int i=0; i<n; i++) {
  scanf ("%d", &arr[i]);
}

for (int i=0; i<n; i++) {
   if (arr[i] %2 != 0) {
     arr[i] = arr[i] * arr[i];
   }
}

printf ("Modified array (odd number squared): \n");
  for (int i=0; i<n; i++) {
  printf ("%d", arr[i]);
  }
return 0;
}
```

Enter number of elements: 5
Enter 5 numbers:
1 2 3 4 5
Modified array (odd number Squared)
1 2 9 4 25

Extra

a)
```
#include <stdio.h>
int main ()
{
   int i , j;
for (i=0 ; i<5 ; itt)
{
  for (j=0 ; j<i ; jtt )
{
     printf( " * ");
   }
   printf ("\n");
}
```

output:

```
*
**
***
****
*****
```

b) 1
12
123
1234

```
#include <stdio.h>
 int main ()
{
  int i, j;
for (i=1; i<=4; i++)
{
 for( j=1 ; j <=i ; j++)
 printf ("%d ", j);
  printf ("\n");
 }
return 0;
}
```

Output -

1
1 2
1 2 3
1 2 3 4

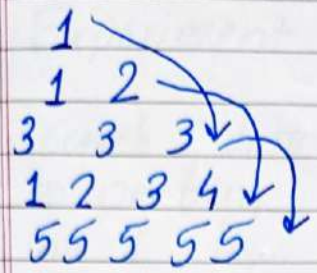
c) 1
2 2
3 3 3
4 4 4 4

```
#include <stdio.h>
int main ()
{
    for (int i = 0; i<5 ; i++)
  {
   for (int j=0; i<5 ; j++)
   {
    for (int j=0 ; j <i ; j++)
      printf ("%d", i);
      printf ("\n");
   }
  return 0;
  }
```

output:

1
2 2
3 3 3
4 4 4 4

d)




```
#include <stdio.h>
int main ()
{
 for (int i=1; i<=5; i++)
 {
    for (int j=1; j<= i; j++)
  {
   if (i%2 ==0)
     printf ("%d", j);
  else
     printf ("%d", i);
   }
   printf("\n");
  }
  return 0;
  }
```

Output

```
1
1 2
3 2 3
1 2 3 4
5 5 5 5 5
```

e)
```
*
# #
$ $ $
? ? ? ?
```

```
#include <stdio.h>
 int main()

{
   for (int i =1;  i<=4 ; i++)
 {
  for (int j=1;  j<=i ,j++)
  {
   if( i==2)
   printf("#");
  else if (i==3)
     printf(" $");
else if(i==4)
   printf(" ?");
  }
  printf("\n");
   }
return 0;
 }
```

Output:

```
*
# #
$ $ $
? ? ? ?
```

Date
Page

★ Experiment 2:

1(a) Search data using linear search consider the quantities to perform using 56, 36, 89, 57, 1, 0, 67, 54. Search no 1 then 55

```
#include <stdio.h>
int main ()
{
   int n[] = {56, 36 89, 57, 1, 0, 67, 54}
   int i, x, f;
   printf ("Array");
   for (i=0; i<7, i++);
   printf ("Enter number to search ");
   scanf ("%d", &x);
   f=0;
   for (i=0; i<7; i++
{
  if (x==n[i])
  {
  printf ("Number at index %d", i);
   f=1;
   break;
   }
}
  if (f==0)
   printf ("Number not found");
  return 0;
  }
```

Output

Array: 56 30 89 57 10 67 54

Enter no. to search: 1
Number found at index 4
Enter number to search: 55
Number not found

2) Search data using binary search

```
include <stdio.h>
int Binary search (int a[], int s, int t)
{
   int i=0, ri= s-1;
   while(l<=ri)
   {
      int m = l+(ri-l)/2;
      if(a[m] == t)
        return m;
      else if (a[m] < t)
        i=m+1;
      else
        ri=m-1;
   }
   return -1;
}
```

```
int main()
    int s, t,
    printf("Enter no. of element
        (sorted array): ");
    int a[5];
    printf("Enter %d sorted elements:",s);
    for (int i=0 ,i<s, i++)
        scanf("%d",&a[i]);
    printf("Enter target element: ");
    scanf("%d", &t);
    int r = binary Search (as, t);
    if(r] == -1)
        printf("Element %d found at index
            %d\n", t, r);
    else
        printf("Element not found");
    return 0;
    }
```

output =

Enter the no. of elements
: 5
Enter 5 sorted array:
2
3
5
7
9

Enter the traget element to search: 5
Element 5 found at index 2

3) Difference between linear and binary search

Linear search	Binary search
i> Checks all the elements of array	Divides array and searches in h
ii> works on both sorted and unsorted array	works only on sorted array
iii> Sequential scan	Logarithmic time
iv> Simple and straight forward	slightly more complex
v) slower on large database	Much faster on larger data base

4) Limitations of linear search in terms of time complexity.

i> Inefficient for larger Data set:
Linear search consumes more time as it checks each and every element as the array may be big and it also depends on array data size of data types. Hence it is not suitable for larger arrays.

ii> No early stopping.
If the element is at the end of the array or not present in the array then the scan will not end early and use max time.

iii) No advantage for sorted data:
Even if the data is sorted, the linear search does not exploite this advantage.

iv) Scalability issues -
If the data contains millions or billions of data, linear search becomes infeasible due to the time being directly proportional to the elements.

5/8/25

1> WAP in C to copy the elements of one array into another array in reverse order.

```
#include <stdio.h>
int main ()
{
   int a[100], r[100], n;
   printf ("Enter the number of elements : ");
   scanf ("%d", &n);
   printf ("Enter %d elements : \n ", n);
   for ( int i = 0 ; i<n ; i++)
   {
     scanf ("%d", & a[i]);
   }
   for (int i = 0 ; i<n ; i++)
   {
     r[i] = a [n - 1 - i]
   }
    print ("Reversed array : \n ");
    for (int i = 0 ; i<n ; i++)
    {
      printf ("%d", r[i]);
    }
    return 0;
}
```

Output -
Enter the no. of elements : 5
Enter 5 elements : 1, 2, 3, 4, 5
Reversed array : 5, 4, 3, 2, 1

Algorithm.

step1: Start
step2: Input variable 'a' for array, r to store reversed array, n to input no. of elements and i for forloop
step3: Input no. of elements.
step4: use for loop to input elements.
step5: use forloop to reverse the elements using r.
step6: Print the reversed array.
step7: stop.

2> WAP to count total no of duplicate elements in array.

```
#include <stdio.h>
int main ()
{
  int a[50], n, c=0;
  printf("Enter no of elements: ");
  scanf("%d", &n);
  printf("Enter %d elements :\n", n);
  for (int i=0; i<n; i++)
    scanf("%d", &a[i]);
  for (int i=0; i<n; i++)
  {
  for (int j=i+1; j<n; j++)
  {
    if (a[i]==a[j])
    c++;
    break;
```

```
    }
  }
}
printf("Total duplicate elements : %d",
       c);
  return 0;
}
```

output -

Enter no of elements: 5
Enter 5 elements:
1, 2, 7, 9, 2

Total duplicate element: 1

Algorithm:

Step1: Start
Step2: Declare a for array, n for no of elements in array, c for count of duplicate elements.
Step3: Input the no. of elements.
Step4: Print the statements to ask user to input elements
Step5: Input elements for the array
Step6: use no for loops use if connection to check is a[i] = a[j] if true.
Step7: Printc (total no of duplicate elements)
Step8: Stop

3> WAP in C to print all unique elements in an array Appear once in a way

```
#include <stdio.h>
int main ()
{
  int a [50] , n, u;
  printf ("Enter no of elements: ");
  scanf ("%d", &n);
  printf("Enter %d elements : ", n);
   for (int i= 0; i<n; i++)
      scanf("%d", &a [i]);

   printf ("Unique elements are : ");
   for (int i = 0; i<n; i++)
   {
     for (int j = 0 ; j <n; j++)
     {
        u = 0;
        break;
     }
   }
   if (u)
     printf ("%d", a [i]);
   }
   return 0;
   }
```

Output :-
Enter number of elements : 3
Enter 3 elements : 1, 2, 1

Unique elements are: 2

Algorithm:
Step1: Start
Step2: Declare a for array, n for no. of elements a for unique i & j for loop
Step3: Enter no. of elements.
Step4: Input elements using for loop
Step5: Check if they are unique
Step6: Print array elements.
Step7: End.

4) WAP to seperate odd & even integers into seperate arrays.

```
#include <stdio.h>
int main ()
{
   int a[50], e[50], o[50];
    int n, ev=0, od=0;
    printf("Enter no of elements : ");
    scanf("%d", &n);
    printf("Enter %d elements:\n", n);
    for (int i=0; i<n; i++)
    scanf("%d", &a[i]);

    for (int i=0; i<n; i++)
    {
     if(a[i]%2==0)
        e[ev++]=a[i];
      else
        o[od++] =a[i];
      }
    printf("Even elements:\n");
      for (int i=0; i<ev; i++)
         printf("%d", e[i]);
      printf("odd elements\n");
      for(int i=0; i<od; i++)
      printf("%d", o[i]);
     return 0;
     }
```

Output:

```
Enter no. of elements : 5
Enter 5 elements -
1
2
3
4
5
Even elements: 2,4
odd elements: 1,3,5
```

5) WAP to find second smallest element in given array.

```
#include <stdio.h>
int main ()
{
  int a[50], n;
  int f = INT_MAX, s = INT_MAX
  printf ("Enter no. of elements ");
  scanf ("%d", &n);
  printf ("Enter %d elements", n);
  for (int i=0; i<n; i++)
  {
    scanf ("%d", &a[i]);
    if (a[i] < f
  {
      s=f
      f=a[i];
```

```
}
else if (a[i]<s && a[i]!=f)
{
   s=a[i];
}
}
if (s==INT_MAX)
   printf("No smallest element.");
else
   printf("second smallest: %d\n",s);
return0;
}
```

Output

```
Enter no of elements:5
Enter 5 elements: 1
 2
 3
 4
 5
Second smallest:2
```

6> WAP to count total no of words in a string.

```
#include <stdio.h>
#include <string.h>
int main ()
{
 char S[100]
 int c =0, iw =0
 printf ("Enter a string:");
 fgets (S, size of (S), stdin);

 for (int i=0 ; S[i] != '\0'; i++)
 {
   iw= 0;
 }
else if (iw ==0)
 {
   iw = 1;
 c++;
 }
 }
printf ("Total no of words: %d \n", c);
return 0;
}
```

Output:
Enter a string king
No. of words: 1

28/7/25

Expt - III

1) Sort elements in ascending order using Bubble Sort:

```
# include <stdio.h>
int main ()
{
 int a[ ] = { 5, 1, 4, 2, 8 };
 int n = size of (a) / size of (a [6]);
 int i, j, t;
 printf ("original array : ");
 for (i=0; i<n; i++)
      printf ("%d", a[i]);

 for (i=0 ; i<n-1 ; i++)
 {
   for (j=0; j<n-i-1 ; j++)
   {
     if (a[j] > a[j+1])
     {
       t=a[j]
       a[j] = a[j+1];
       a[j+1] = t;
     }
   }
 }
```

```
printf ("Sorted array : ");
for (i=0; i<n; i++)
  printf ("%d", a[i]);

return 0;
}
```

output:
original array:
5 1 4 2 8
Sorted array
1 2 4 5 8

2) Sort elements in descending order using selection sort.

```
#include <stdio.h>
int main ()
{
 int a[] = {5, 1, 4, 2, 8};
 int n = sizeof(a) / sizeof(a[0]);
 int i, j, m, t;

printf ("original array: ");
 for (i = 0; i<n; i++)
   printf ("%d ", a[i]);

 for (i = 0; i<n; i++)
 {
```

```
m=i;
for(j= i+1; j<n;j++)
{
 if (a[j]> a [m])
  {
   m=j;
  }
}
  t=a [i];
a[i] = a [m];
a[m] = t;
 }
 printf (" sorted array indescending order");
 for(i=0; i<n; i++)
    printf( "%d", a[i]);
 return 0;
}
```

output :-

original array

5 1 4 28

sorted array in descending order

28 5 4 1

3) Find the no. of comparisons required in bubble sort method of the following data having 5 numbers: 100, 200, 300, 400, 500

Given numbers: 100, 200, 300, 400, 500

100 | 200 | 300 400 500

100 | 200 | 300 | 400 500

100 200 | 300 | 400 | 500

100 200 300 | 400 | 500 |

Hence only 1 pass and 4 comparisons take place as the given nos are already sorted.

1- Strat with list of nos
2- compare each pair of adjacent elements and swap if the first element is greater than second element. Continue comparing and sorting.
3- After each full pass the the largest unsorted element moves to the end of array.
4- Repeat the process of 2 step on the unsorted array.
5- Stop when all elements are in the correct order

4) Sort the given array in selection sort for ascending order and show diagrammatic representation of every iteration of the loop : 500, -20, 30, 14, 50

500 -20 30 14 50 } 1st

-20 500 30 14 50 } 2nd

-20 30 500 14 50

-20 14 500 30 50 } 3rd

-20 14 30 500 50 } 4th

-20 14 30 50 500 } 5th

Hence only 4 passes and 5 comparisons take place

Algorithm:

1- set min to 0
2- search min element of the arr
3- swap value of the location of min
4- Increment min to point the next element location
5- Repeat till sorted
6 - Stop

Flow chart

Experiment - IV

a) sort elements in ascending order using insertion sort.

```
#include <stdio.h>
int main()
{
 int a[] = {7, 3, 5, 1, 9, 8, 4, 6};
 int n = sizeof(a)/sizeof(a[0]);
 int i, j, k;

 for (i=1; i<n; i++)
 {
  k = a[i];
  j = i-1;
  while( j >=0 && a[j]>k)
  {
    a[j+1] = a[j];
     j = j-1;
   }
 a[j+1] = k;
 if(i==2)
 {
    printf("Array:")
  for (int i=0; i<n; i++)
    printf("%d", a[i]);
   printf("\n");
   }
 }
```

```
printf ("Sorted Array:");
 for (i=0; i<n; i++)
   printf("%d", a[i]);
  return 0;
}
```

output:

Array

3 5 7 1 9 8 6

Sorted array:

1 3 5 6 7 8 9

b> sort element in ascending order using radix sort.

```
#include <stdio.h>
 int getMax (int a[], int n)
 {
     int m = a[0];
     for (int i = 1; i<n ; i++)
     {
       if (a[i] > m)
       m = a[i];
       return 0;
       }
  }
void sort (int a[], int n, int exp)
{
```

```
int o[50], c[10]={0}

for(int i=0; i<n; i++)
  c[(a[i]/exp)%10]++;
for(int i=1; i<10; i++)
  c[i] += c[i-1];

for(int i=n-1; i>=0; i--)
{
  o[c[(a[i]/exp)%10]-1] =
    a[i]
  c[(a[i]/exp)%10]--;
}
for(int i=0; i<n; i++)
  a[i]=o[i];
}

int main()
{
int a[] = {7,3,5,1,9,8,9,4,6}
int n = sizeof(a)/sizeof(a[0]);

radixSort(a,n);
printf("Sorted Array: ");
for(int i=0; i<n; i++)
  printf("%d", a[i]);
  return 0;
}
```

Output

Sorted Array

1 3 4 5 6 7 8 9

c What is the output of insertion sort after the 2nd interation given the following sequence of no.s : 7, 3, 5, 1, 9, 8, 4, 6

7 3 5 1 9 8 4 6

3 7 5 1 9 8 4 6

3 5 7 1 9 8 4 6

1 3 5 7 9 8 4 6

1 3 4 5 7 9 8 4 6

1 3 4 5 6 7 9 8

1 3 4 5 6 7 8 9

Sort the following no.s using Radix Sort:-

100 225 390 4130 956 99 5431

Pass 1	0	1	2	3	4	5	6	7	8
0100	0100								
0225						0225			
0390	0390								
4130	4130						0956		
0956									
0099									
5431		5431							

Pass 2)	0	1	2	3	4	5	6	7	8
0100		0100							
0225			0225						
4130		4130							
5431					5431				
0956									
0390									
0099	0099								

~~Pass 3)~~

	0	1	2	3	4	5	6	7	8	9
Pass 1										
0100		0100								
0225			0225							
4130		4130								
5431					5431					
0956										
0340										
0099	0099									

Pass 2	0	1	2	3	4	5	6	7	8	9
0099	0099									
0100	0100									
4130					4130					
0225	0225									
0340	0340									
5431						5431				
0956	0956									

Sorted arr [009, 010, 025, 039, 056, 136, 145]

Pass1)	0	1	2	3	4	5	6	7	8	9
025						025				
006							006			
099										099
145						145				
239										239
020	020									
018								018		

Pass2)	0	1	2	3	4	5	6	7	8	9
020			020							
025			028							
145					145					
006	006									
018		018								
099										099
239										

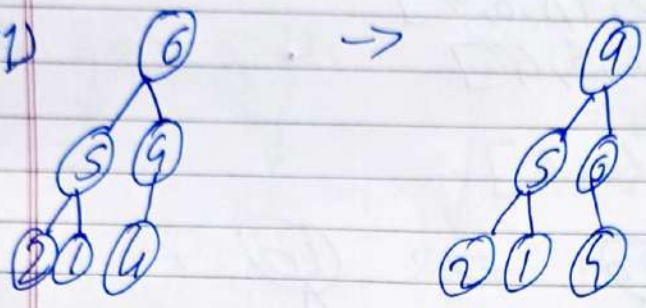
Pass3)	0	1	2	3	4	5	6	7	8	9
006	006									
018	018									
020	020									
~~239~~ 025	025		239							
~~145~~ 239	239	~~145~~								
~~099~~ 145	~~099~~	145								
099	099									

Sorted arr [006, 018, 020, 025, 099, 145, 239]

Q4) Sort the following element using heap sort :-

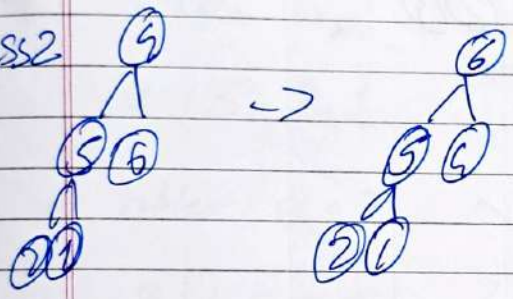
arr[6, 5, 9, 2, 1, 4]

Pass1)



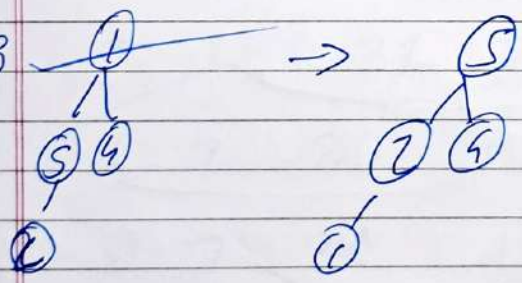
arr[9, 5, 6, 2, 1, 4]
arr[4, 5, 6, 2, 1, 9]

Pass2



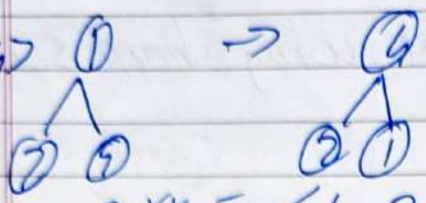
arr = [6, 5, 4, 2, 1, 9]
arr = [1, 5, 4, 2, 6, 9]

Pass3



arr = [5, 2, 4, 1, 6, 9]
arr = [1, 2, 4, 5, 6, 9]

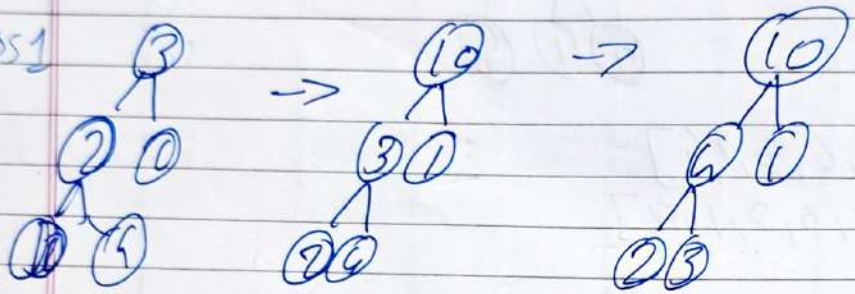
Pass 4)



arr = [4, 2, 1, 5, 6, 9]
arr = [1, 2, 4, 5, 6, 9]

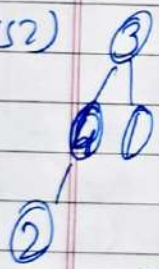
2) arr [3, 2, 1, 10, 4]

Pass 1



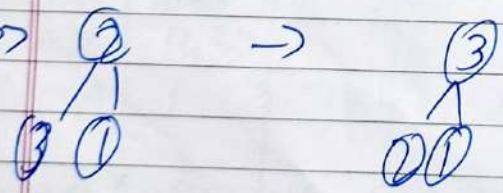
arr = [10, 4, 1, 2, 3]
arr = [3, 4, 1, 2, 10]

Pass 2)



arr [4, 3, 1, 2, 10]
arr [2, 3, 1, 4, 10]

Pass 3)

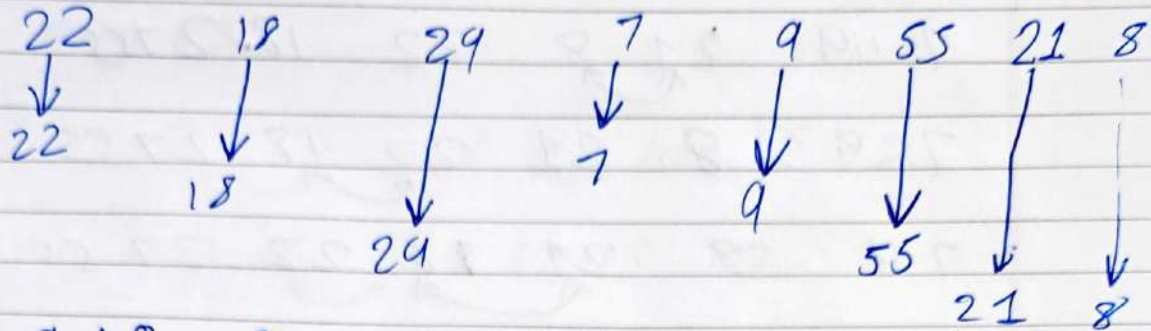


arr [3, 2, 1, 4, 10]
arr [1, 2, 3, 4, 10]

Q5] Sort the following using shell sort

arr [22, 18, 29, 7, 9, 55, 21, 8]

Pass 1



9 18 29 7 22 55 21 8

9 18 29 7 22 55 21 8

9 18 21 7 22 55 29 8

after gap

9 18 21 7 22 55 29 8

Taking gap = 2

9 18 21 7 22 55 29 8

9 7 21 18 22 55 29 8

9 7 21 18 22 8 29 55

9 7 21 8 22 18 29 55

Gap = 1

9 7 21 8 22 18 27 55

7 9 21 8 22 18 27 55

7 9 8 21 22 18 27 55

7 9 8 21 22 18 27 55

7 9 8 21 22 27 55

7 8 9 21 22 27 55

∴ Sorted array : 7, 8, 9, 18, 21, 22, 29, 55

(i) 3, 10, 15, 12, 1, 5, 2, 6

array 12 = 4

3 10 15 12 1 5 26

1 10 15 12 3 5 26

1 5 15 12 3 10 26

1 5 15 6 3 10 15 12

after gap: 1, 5, 2, 6, 3, 10, 15, 12

gap = 2

15 2 6 3 10 15 12

gap = 3.

15 2 6 3 10 15 12

1 2 5 6 3 10 15 12

1 2 5 3 6 10 15 12

1 2 3 5 6 10 15 12

1 2 3 5 6 10 12 15

∴ Sorted array:

1, 2, 3, 5, 6, 10, 12, 15

25/8/25 (complete Radix sort)

★ Experiment 5:

Singly linked list using menu:

→
```
#include<stdio.h>
#include<stdlib.h>
struct node {
 int d;
struct node*n;
};
struct node * n = NULL;
void ibc (int x)
{
    struct node*n = NULL;
    struct node*t = malloc (size of (
    struct node ));
    t ->d = x;
    t ->n = n;
    n = t;
}
void ie (int x)
{
   struct node * t = malloc (sizeof (
    struct node))
   t->d = x;
   t ->n = NULL;
   if (!n)
{
     n=t;
   }
else
```

```
{
  struct node *p = h;
  while (p->n) p = p->n;
    p->n = t;
  }
}
void del (int x)
{
  struct node *p = h, *q = NULL
  while (p && p->d != x)
{
  q = p;
  p = p->n;
}
if (!p)  return;
if (!q)
        h = h->n;
  else
    q->n = p->n;
    free(p);
  }
void s (int x)
{
  struct node *p = h, *q = NULL;
  while (p)
    {
      if (p->d == x)
    {
        print ("Found")
        return;
```

```
}
  p=p->n;
 }
 printf("Not found");
}
 void dis()
{
 struct node *p=h;
  while(p)
 {
  printf("%d ", p->d);
     p=p->n;
    }
    printf("\n");
  }
int main () {
  int c, x;
   for(;;)
   {
     printf("1. IB 2. IE 3. DEL 4. SEARCH 5. Display
            6. Exit\n");
     scanf("%d", &c);
      if(c==6)
       break;
      switch(c)
      {
        case 1:
         scanf("%d", &x);
           ib(x)
          break;
```

```
case 2:
        scanf ("%d", &x);
          ie (x)
          break;

case 3:
        scanf ("%d", &x);
        del (x)
        break;
case 4:
        scanf ("%d", &x);
        s(x)
        break;
case 5:
        dis();
        break;
      }
  }
}
```

output: 1. IB 2. IE 3. DEL 4. SEARCHS. DISP
6. EXIT

6

10/11

Experiment 6

Doubly linked list menu driven program

```
#include <stdio.h>
#include <stdlib.h>
  struct node
{
   int d;
   struct node*p, *n;
};
struct node *h = NULL;
 void ib (int x)
{
   struct node*t = malloc (sizeof (struct node));
      t->d = x;
      t->p = NULL;
      t->n = h;
      if(h) h->p = t;
       h=t;
  }
 void ie (int x)
 {
      struct node* t=malloc (size of (struct
      node)) *q = h;
       t->d = X
       t->n = NULL;
       if(!h)
     {
```

```
t->p = NULL;
   h=t;
   return;
}
 while (q->n)
{
   q=q->n;
   q->n=t;
   t->p=q;
  }
}
void del (int x)
{
  struct node *q=h;
  while (q && q->d!=x)
       q=q->n;
   if(!q)
   return;
  if(q->p)
    q->p->n = q->n;
else
    h=q->n;
    if (q->n)
      q->n->p = q->p;
     free(q);
  }
  int s (int x)
{
    struct node *q=n;
    int pos = 1;
    while (q)
```

```
{
    if(q->d==x)
        return pos;
        q=q->n;
    pos++;
    }
    return 0;
}
void dis()
{
 struct node *q=n;
  if(!q)
  {
     printf("Empty");
      return 0;
     }
  while(q)
  {
    Printf("%d", q->d);
      q=q->n;
    }
    Printf("\n")
}
int main()
{
  int ch, x, y;
  while(1)
  {
Printf("1. IB 2. IE 3. DEL 4. SEA 5. DIS GEXIT\n")
Scanf("%d", &ch);
switch(ch) {
```

```
case 1:
    scanf ("%d", &x);
     ib(x)
    break;
 case2:
      scanf("%d", &x);
       ic(x)
      break;
 case3:
      scanf("%d", &x);
   del(x)
    break;
case 4:
      scanf("%d", &x);
     r=s (x)
    if(r)
    printf ("Found");
   else
     printf("Not found");
   break;
case 5:
dis();
  break;
  case 6 :
  exit(0);
   }
  }
}
```

Output:

1) INS 2) DEL 3. SEA 4. DIS 5. EXIT
6.

10/11

Experiment 7

Stack menu driven program for push pop display & exit

```
#include <stdio.h>
#include <stdlib.h>
#define size 5
void push();
void pop();
void display();
int stack[size];
int top = -1;
int main()
{
    int ch;
    printf("1. PUSH\n2. POP\n3. Display\n4. EXIT");
    do
    {
        printf("Enter choice: ");
        scanf("%d", &ch);
        switch(ch)
        {
            case 1:
                push();
                break;
            case 2:
                pop();
                break;
            case 3:
                display();
```

```
        break;
        case 4:
          exit(0);
          break;
        }
}
void pop()
{
    if(top==-1)
      printf("Stack is empty");
    else
    {
      printf("Deleted element is = %d",
                stack[top]);
              top--;
          }
    }
void push()
{
    int v;
    if(top==max-1)
      printf("Overflow");
    else
{
    printf("Enter element :");
    scanf("%d", &v);
    stack[++top] = v;
    }
  void display()
{
```

```
if (top == -1)
    printf ("Empty");
  else
{
    printf ("Elements are: ");
    for (int i=0 , i<= top; i++)
    {
       printf ("%d", stack [i]);
    }
  printf ("\n");
  }
}
```

Output

1. PUSH 2. POP 3. DISPLAY 4. EXIT

Enter choice: 4

10/11

★ Experiment-8

i) Convert infix to postfix expression:

```
#include <stdio.h>
#include <ctype.h>
 char stack [100],
int top = -1;
void push (char x)
 {  stack [++ top] = x;
   }
 char pop ()
 {if (top == -1)
     return -1;
  else
     return stack [top---]
  }
  int priority (char x)
  { if (x == '(')
      return 0;
      if (x == '+' || x == '-')
      return 1;
      if(x == '*' || x == '/')
      return 2;
      return 0;
   }
   int main (){
   char exp[100];
    char *e, x;
    printf ("Enter the expression");
```

```
scanf("%s", &exp);
printf("\n");
  e=exp;
while (e != '\0'){
 if(isalnum(*e))
printf("%c",*e);
 else if( *e == '(')
 push(*e);
else if(*e == ')')
 {
while( (x=pop()) != '(')
printf("%c",x);
    }
else {
while(priority stack[top] >= priority(*e))
printf("%d",pop());
 push(*e);
  }
e++;
while(top != -1) {
 printf("%c",pop());
    }
  return 0;
    }
```

Output:

Enter infix expression: A+(B*C-(D/E^F)*G*H

postfix expression= ABCDEF^/G*H*+

q2) Input	Stack	o/p
1) H	−	H
*	*	H
)	*)	H
G	*)	HG
*	*)*	HG
)	*) *)	HG
F	*) *)	HGF
^	*) *) ^	HGF
E	*) *) ^	HGFE
/	*) *) ^ /	HGFE^
D	*) *) /	HGFE^D
C	*) *	HGFE^D/*
−	*) * −	HGFE^D/*C
(	*) * − (	HGFE^D/*C
*	*) * − (*	HGFE^D/*C
B	*	HGFE^D/*CB+-
C	+	HGFE^D/*CB+-^
* +	+	HGFE^D/*CB+-^+
A		HGFE^D/*CB+-^+*
2) 2		2
4		2, 4
3	4 2	16
1		16, 1
2		16, 1, 2
4		16, 1, 2, 4
3	4 2	16, 1, 2, 8
2		16, 1, 2, 2
1		16, 1, 2, 2, 1
+	2 1	16, 1, 2, 3
3	3	
4	6	

P.T.O

*	3	2	16, 1
-	6	1	16, 5
+	5	16	21

Q3. Evaluate postfix expression:

```
#include <stdio.h>
# include <ctype>
# include <stdlib.h>
# define size 40
  int pop ();
 void push(int);
  Char postfix [size];
 int stack [SIZE];
 int main (){
 int i, a, b, result PEval;
  char ch;
 for (i= 0; i<SIZE; i++){
 stack [i] = -1;
  }
scanf printf ("%s", postfix);
print ("Enter a postfix expression");
 for(i=0, postfix [i] = '\0'; i++ ){
   ch = postfix [i]
   if(is digit(ch)){
    push (ch-'0');
      }
else if(ch='+'
```

```
b = pop();
a = pop();
case '+':
result = a+b;
break;
case '-':
result = a-b;
break;
case '/':
result = a/b;
break;
case '*':
result = a*b;
break;
}
push(result);
}
}
{ val = pop();
printf("In the feature evaluation is %d\n",
return 0;
}
void push(int n) {
int pop() {
int n;
if(top > -1) {
n = stack[top];
stack[top--] = -1;
return n;
} else {
printf("stack is empty:\n");
exit(-1);
}
}
```

Experiment - 9

9 WAP in C to perform primitive operations on linear queue.

```
#include <stdio.h>
#include <stdlib.h>
# define size 5
void insert ();
void delete ();
void display ();
int queue [size];
int front = -1, rear = -1
int main ();
{
  int ch;
printf ("1. Insert\n2. Delete\n 3. Display\n4. Exit\n
do{
printf ("\nEnter your choice ");
scanf ("%d", &ch);
switch (ch) {
case 1: insert ();
break;
case 2: delete ();
break;
case 3: Display(); break;
case 4: exit
  break;
} while (ch!=4);
  return 0;
}
```

```
void insert()
{
int element;
if(rear==size-1){
printf("\n queue is full...");
}else{
printf("Enter element to insert :");
Scanf("%d", &element);
if(front==-1) front = 0;
rear++;
queue[rear=element;
}
}
void delete()
{
if(front==-1||front>rear){
printf("\n queue is empty");
}else{
printf("\n Deleted element is=%d",
queue[first]);
front++;
}
}
Void display()
{
if(front==-1||front>rear){
print("\n queue is empty...");
}else{
print("\n queue element are:\n")
for(int i<front;i<=rear;i++){
print("\t%d", queue[i]);
}
}
```

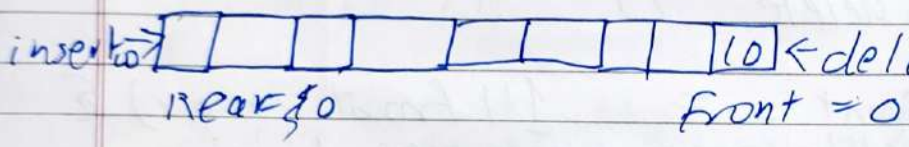
Output:

1. Insert
2. Delete
3. Display
4. Exit

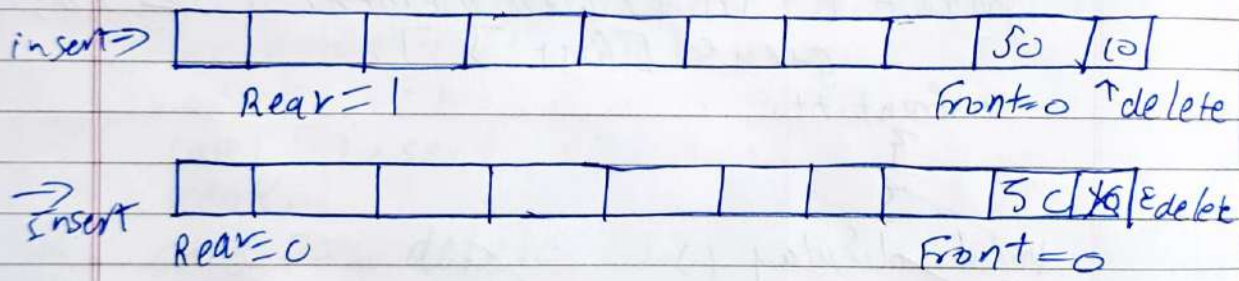
Enter your choice: 4

Q2. Perform.
Insert 10, Insert 50, Delete, Insert 102, Insert 20, Delete, Insert 25, Insert 200

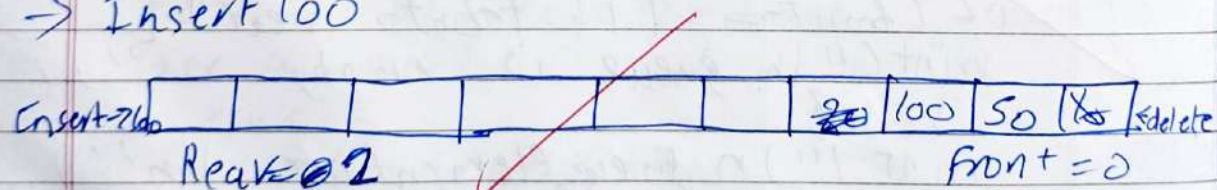
→ Insert (10)



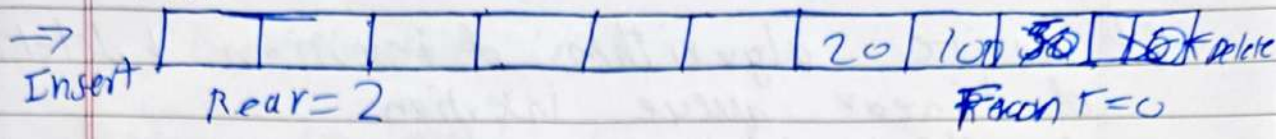
→ Insert (50)



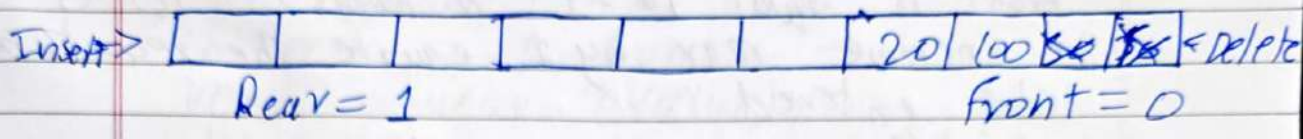
→ Insert 100



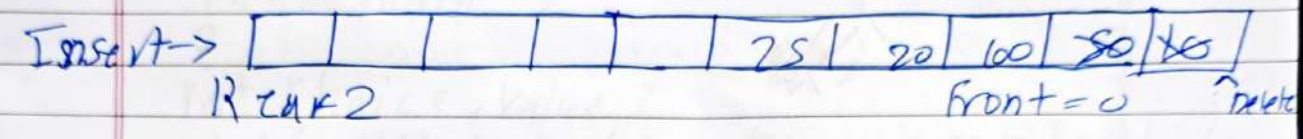
- ~~Delete~~ Insert 20



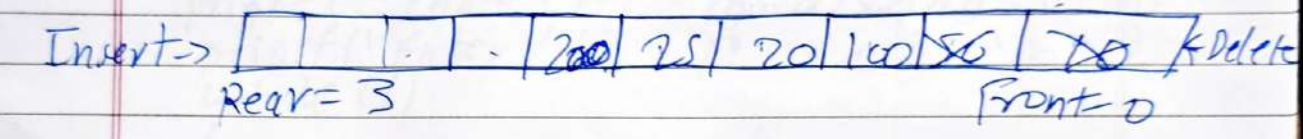
- Delete



- Insert (25)



Insert (200)



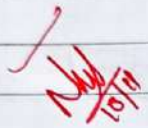
* Theorey question:

Q1] Explain the concept of priority queue.

A priority queue is a special type of queue where each element is assigned a priority and elements are served based on their priority or just their arrival order.

Q2. write algorithm of insertion & deletion of linear queue insertion

1. check if the queue is in overflow condition
2. If no overflow condition check if front is equal to -1 & change it to 0
3. Increase rear by 1 equate the rear to the value entered.
4. Print the value equeued.



etc

Experiment 10

```
#include <stdio.h>
#define MAX 5
  int queue[MAX]
int rear = -1, front = -1
void enqueue (value);
 void dequeue ()
 void display ();
 int main ()
 {
 int choice, value;
 printf ("Enter '1' to enqueue: \n");
 printf ("Enter '2' to dequeue: \n");
 printf ("Enter '3' to display: \n");
 printf ("Enter '4' to exit: \n");
 while (1)
 {
 printf ("Enter choice: ");
 scanf ("%d" &choice);
   switch (choice)
  {
case 1: printf ("Enter value: ")
        scanf ("%d" &value);
        enqueue(value);
        break;
case 2: dequeue ()
        break;
case 3: display()
        break;
```

```
case 4 : printf("Exiting !\n");
         return 0;
      default : print ("Invalid input !\n");
    }
  }
}
void enque (int n)
{
  if (rear+1) % MAX == front)
  {
  print ("queue overflow !\n");
  }
  else
  {
    if (front == -1)
    {
      front = 0;
    }
    rear = (rear+1) % MAX;
    queue [rear] = n
    printf ("%d enqueued !\n", n);
  }
}
void dequeue ()
{
  if (front == -1)
  {
    printf ("queue underflow !\n");
  }
  else
  {
```

```
printf("%d deleted from queue\n",
        queue[front]);
    if (front == rear)
    {
        front = rear = -1;
    }
    else
    {
    front = (front + 1) % MAX;
    }
  }
}
    void display()
    if(front == -1)
    {
      printf("queue Empty!\n");
    }
    else
    {
      print("queue Elements:\n");
      int i=front;
    while(1)
    {
      printf("%d ", queue[i]);
      if (i == rear) break;
          i = (i+1) % MAX;
      }
      printf("\n");
    }
}
```

Theorey questions

Q1. write algorithm for ~~what are advantages~~ insertion & deletion operation on circular queue insertion algo.

1) Check for overflow condition in queue.
2) Else if front and rear are equal to -1 change it to 0
3) Else increase rear value by 1
4) Print value inserted

deletion algorithm
1) check for underflow condition in queue
2) Else delete the value
3) print value deleted

Q2 write & explain application of queue
1. CPU scheduling: process waiting to be executed by the CPU are stored in a ready queue

2. Breadth first search (BFS): It uses queue.

3. OS. Manage access to resources like printers files memory.

Q3. What are advantages of circular queue, show diagrammatic representation.

1) Efficient we use of memory - all array slot can be used
2) No need for shifting element - unlike linear queue
3) Better performance in repeated enqueue Idequeue operation.
4) Practical applications - used in CPU scheduling, Suffering.

initial (empty)

[ ] [ ] [ ] [ ] [ ]

F, R

After enque 10, 20, 30

[10] [20] [30] [ ] [ ]

After dequeue (remove 10)

[ ] [20] [30] [ ] [ ]
F

Enqueue 40, 50

[40] [20] [30] [50] [ ]
R F

Enqueue 60 (wraps around)

[40] [20] [30] [50] [60]

Experiment 11

Q WAP in C to implement all operations on a binary tree; creation, display, deletion, search

```
#include <stdio.h>
#include <stdlib.h>
 struct node {
int data;
struct node *left *right; };
struct node * create Node (int value){
struct node * new N = (struct node*)
 mall oc
(size of (struct node));
new N -> data = value;
New N-> left = new N -> right -> NULL
return new N;
    }
struct node * insert (struct node* root, int
 value){
if (root == NULL) return create Node
  if (value < root -> data)
  root -> left = insrt (root -> left, value);
else
root -> right = insert (root -> right, value);
 return root;
    }
Void inorder (struct node * root) {
if (root == NULL || root -> data == key
return root;
if (key < root -> data)
```

```
return search(root->right, key);
}
struct t *nodedelete Node(struct node*,
int key) {
if (!root) return root;
if (key < root -> data)
root -> right = delete Node(root->right)
else if {
if (! root -> left ) return root -> right
if (! root -> right) return root -> left;
struct node * temp = root -> right;
while (temp -> left ) temp = temp -> left;
root -> data = temp -> data;
root -> right = delete Node (root ->
right temp -> data
}
return root;
}
int main () {
struct node *root = NULL;
int choice, val;
do {
printf("\n1. Insert 2. Display 3. \n search
4. \n Delete 5 \n Exit \n );
scanf(" %d ", &choice);
case 1:
Scanf ("%d", &val)
root = insert (root, val);
break;
case 2:
inord, r (root)
```

```
break;
case 3:
  scanf ("%d" &val);
if search (root, val) printf ("found\n")
else printf ("Not found")
break;
case 4:
scanf ("%d", &val);
root = deleteNode (root, val);
printf ("Deleted (if existed)\n"; break;

case 5:
  break;
default: printf ("Invalid choice:");
    }
        } while (choice != 5);
  return 0;
        }
```

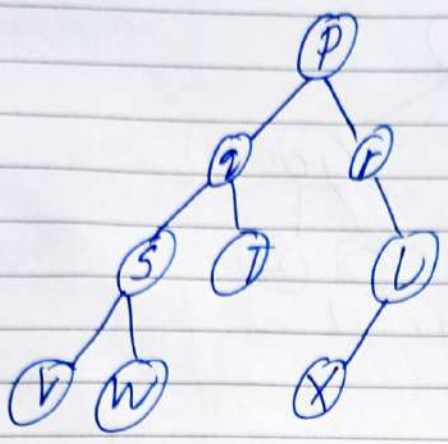
O/P:

1. Insert
2. Display
3. Search
4. Delete
5. Exit

5.

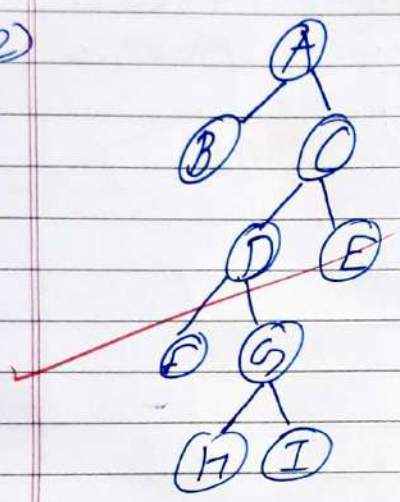
Theorey

1)



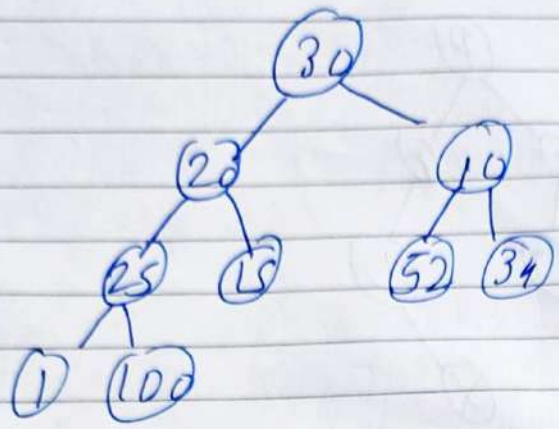
Preorder: P → Q → S → V → W → T → R → U → X
Inorder: V → S → W → Q → T → P → X → U → R
Postorder: V → W → S → T → Q → X → V → R → P

2)



Preorder: A → B → C → D → F → G → H → I → E
Inorder: B → A → F → D → H → G → I → C → E
Postorder: B → F → H → I → G → D → F → C → A

3 30, 20, 10, 25, 15, 52, 34, 1, 100



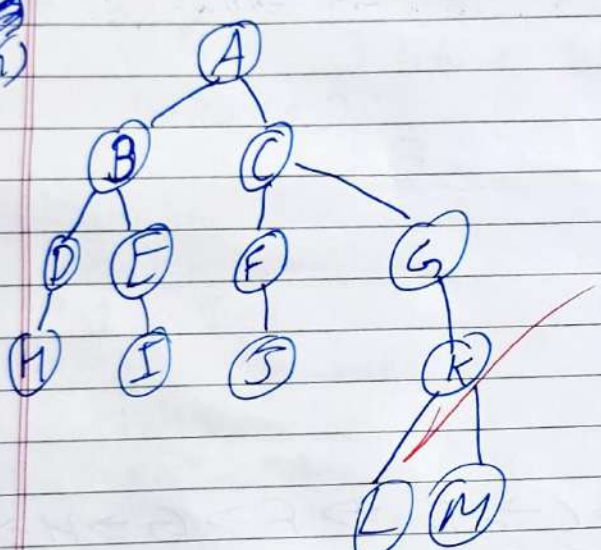
Pre order: 30 → 20 → 10 → 1 - 15 → 25 - 52 → 34 - 1

Inorder → I → 25 → 100 → 20 → 15 → 52 → 10 →

Post order → 1 → 100 → 25 → 15 → 20 → 52 → 34 → 10 →

30

2)

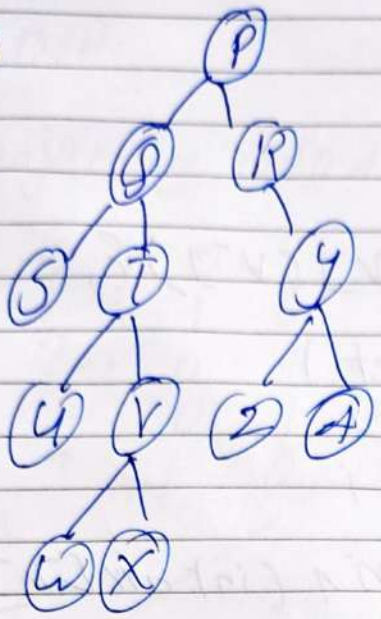


Preorder: A → B → D → H → E → I → C → F → J → G → K → L → M

Inorder: H → D → B → E → I → A → J → F → C → L → K → M → G

Postorder: H → D → I → E → B → J → F → L → M → K → G → C → A

5]



Preorder → P - Q - S - T - U → V - W - X - Y - Z - A

Inorder : S → Q → U → T → W → X → P → R → Z - Y - A

Postorder : S → U → W → X → V → T → Q → Z → A → Y - P

10/11

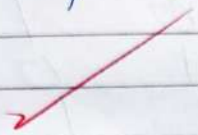
Experiment 12

```
#include<stdio.h>
#define V 5
void init(int arr[V][V]){
 int i, j;
 for (i=0; i<V; i++)
 for(j=0; j<V; j++)
  arr [i] [j] =0;
 }
void printAdjMatrix(int arr[V][V]){
int i, j;
printf("Adjacency matrix:\n");
 for(i=0; i<V; j++){
 printf("%d:",i);
  for (j=0; j<V; j++){
  printf("%d", arr[i][j]);
  }
 printf("\n");
 }
}
int main{
 int adjMatrix[V][V];
 int (adjMatrix);
 insertEdge(adjMatrix,0,1);
insertEdge(adjMatrix,0,4);
insertEdge(adjMatrix,1,2);
insertEdge(adjMatrix,1,3);
printAdjMatrix(adjMatrix);
  return0;}
```

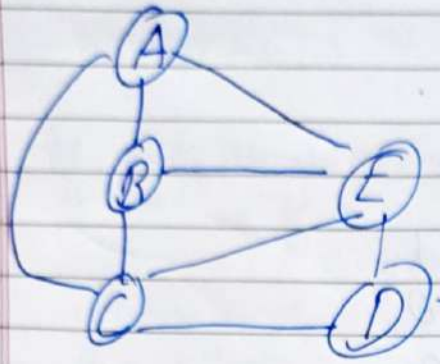
O/P

adjacency Matrix:

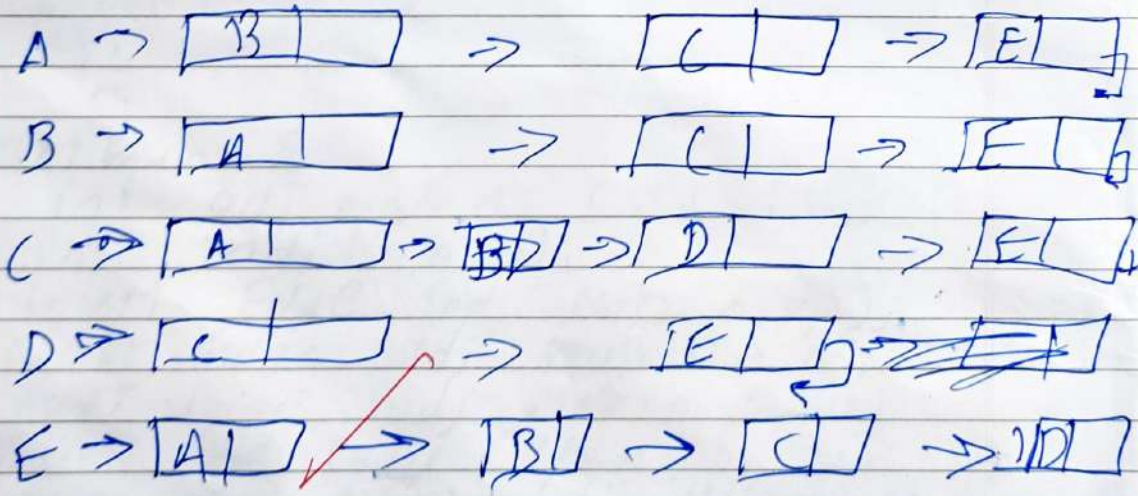
0	:	0	1	0	0	1
1	:	1	0	1	1	0
2	:	0	1	0	0	0
3	:	0	1	0	0	0
4	:	1	0	0	0	0



★

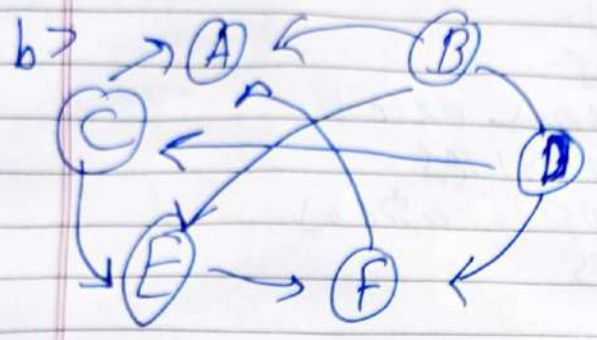


A	0	1	1	0	1
B	1	0	1	0	1
C	1	1	0	1	1
D	0	0	1	0	1
E	0	1	1	1	0



0	1	1	0	1
1	0	1	0	1
1	1	0	1	1
0	0	1	0	1
1	1	1	1	0

b)



*	A	B	C	D	E	F
A	0	0	0	0	0	0
B	1	0	0	1	1	0
C	1	0	0	0	1	0
D	0	0	1	0	0	1
E	0	0	1	0	0	1
F	1	0	0	0	0	1

A → NULL

B → [B] → [D] → [E] ↘

C → [A] → [E] ↘

D → [C] → [F] ↘

E → [C] → [F] ↘

F → [A]

Q2) Applications of graph
In computer science
Google maps
facebook
world wide web
operating systems
mapping systems
Microsoft ~~Systems~~ excel
~~Att~~ social media sites
biochemical applications
bij kasbra algos.

Q3) Explain terminologies.

1) vertex / Node.
→ It is a point in graph where edges connect. It represents an entity/ objects.

2) Edge
→ It is a line that connects two vertices.

3) Adjacency
→ Two vertices are adjacent if they're directly connected by an angle.

3) Path
→ A Path is a sequence of connected vertices. where each successive pair is connected by an angle

4. Directed graph
→ Edges have a specific direction, connection are shown using arrows.

5. Unidirected graph.
→ Edges have no specific direction, it has two-way or mutual relationship.