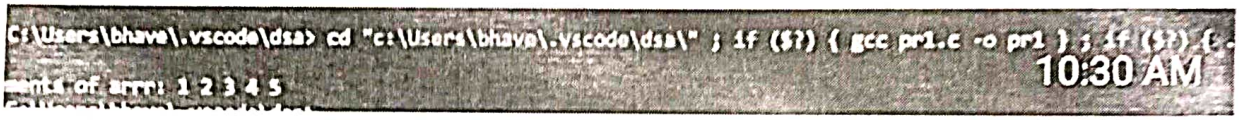
Practical No : 1

Aim : Develop and execute a c program to traverse and display the elements from the given array.



Page No.

Date

Practical No-1

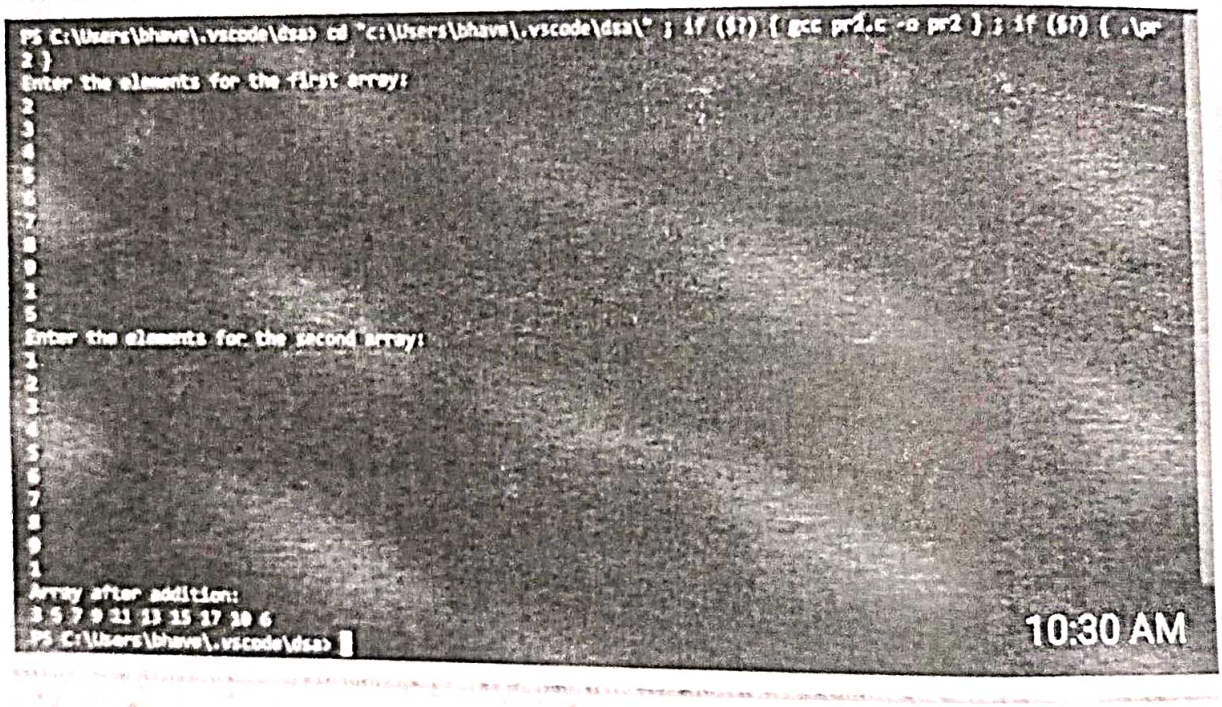
Aim :- Develop and execute a c program to traverse and display the elements from the given array.

Theory :-

Traversing basically means the accessing the each and every element of the array at least once. traversing is usually done to be aware of the data elements which are present in the array.

Program :-

```
#include <stdio.h>
int main()
{
 int arr[] = {1,2,3,4,5};
 int size = size of (arr) / size of (arr[0]);
   // traversity and display the element of the
   array.
   printf("elements of arr:");
    for (int i = 0; i < size; i++)
    {
       printf("%d", arr[i]);
    }
    return 0;
}
```


PS C:\Users\bhave\.vscode\dsa> cd "c:\Users\bhave\.vscode\dsa\" ; if ($?) { gcc pr2.c -o pr2 } ; if ($?) { .\pr
2 }
Enter the elements for the first array:
2
3
4
5
6
7
8
9
1
5
Enter the elements for the second array:
1
2
3
4
5
6
7
8
9
1
Array after addition:
3 5 7 9 11 13 15 17 10 6
PS C:\Users\bhave\.vscode\dsa>
10:30 AM

Page No.

Date

2) Write a program to add two arrays (int) of size 10 and store the result in 3rd array & display ~~in~~ ilan o/p screen.

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

    int arr1[10], arr2[10], result[10];
    printf("enter the elements for the first array");
    for (int i=0; i<10; i++){
        scanf("%d", &arr1[i]);
    }
    printf("Enter the elements for the second array");
    for(int i=0; i<10; i++){
        scanf("%d", &arr2[i]);
    }
    printf("Enter the elements for the second
    for(int i=0; i<10; i++){
        result[i]=arr1[i]+arr2[i];
    }
    printf("Array after addition");
    for(int i=0; i<10; i++){
        printf("%d", result[i]);
    }
    return 0;
}
```


PS C:\Users\bhave\.vscode\dsa> cd "c:\Users\bhave\.vscode\dsa\" ; if ($?) { gcc pr3.c -o pr3 } ; if ($?) { .\pr
3 }
matrix:
295
394
343
PS C:\Users\bhave\.vscode\dsa>
10:30 AM

3) Write a program to create 2×2 array initialize it and display it on output screen in matrix format.

```
#include <stdio.h>
#include <math.h>

int main(){
    int row=3;
    int column=3;
    int matrix[3][3]={2,3,5}
                     {3,1,4}
                     {2,4,3}
}
printf("matrix\n");
 for(int i=0; i<row; i++){
        for(int j=0; j<column; j++){
        printf("%d\n", matrix[i],[j]);
}
printf("\n");
return 0;
}
```

Page No.

Date

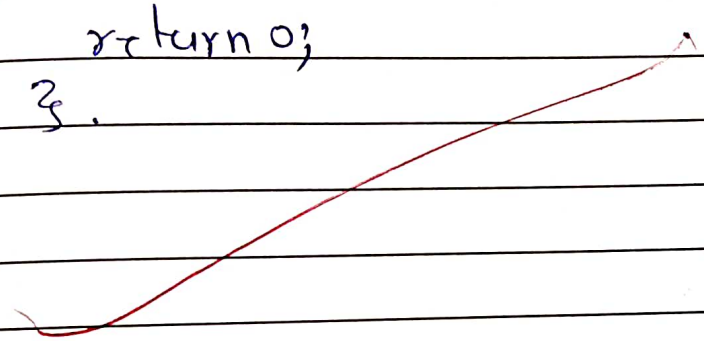
4) Write a program to create two arrays of size 3x3 and perform addition of it. store the result in 3rd array and display it on output screen.

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

    int arr1[3][3] = {
                        {2,1,3},
                        {4,5,6},
                        {7,8,9}
                    };

    Int arr2[3][3] = {
                        {9,8,7},
                        {6,5,4},
                        {3,2,1});
                    };
int result[3][3];
  for(int i=0; i<3; i++)
  {
    For(int j=0; j<3; j++){
          result[i][j] = arr1[i][j] +
                         arr2[i][j];

    }
  }
Printf("Result matrix : \n");
```

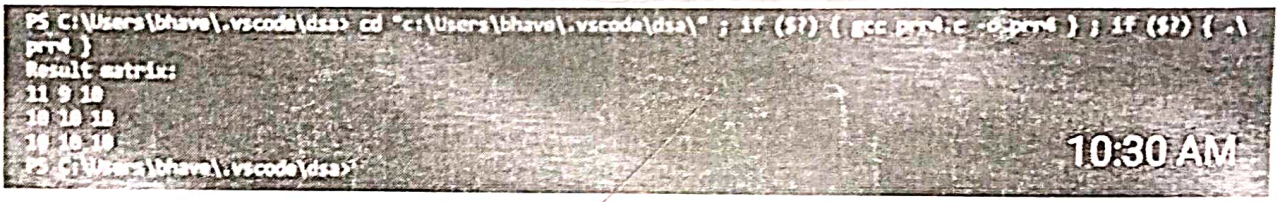
Page No.

Date

```
for (int i=0; i<3; i++){

    for(int j=0; j<3; j++){
        printf("%d", result[i][j]);
    }
    printf("\n");
}
return 0;
}.
```



Teacher's Signature


PS C:\Users\bhave\.vscode\dsa> cd "c:\Users\bhave\.vscode\dsa\" ; if ($?) { gcc prg4.c -o prg4 } ; if ($?) { .\prg4 }
Result matrix:
11 9 10
10 10 10
11 10 10
PS C:\Users\bhave\.vscode\dsa>
10:30 AM

Page No. :
Date :

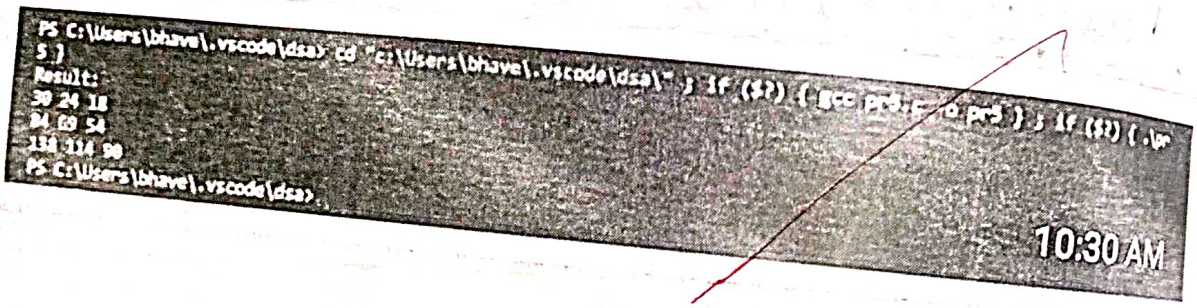
5. Write a program to create two array of size 3x3 perform multiplication of it. store result in 3rd array display it on o/p screen.

```
#include<stdio.h>
    int main()
  { int matrix1[3][3] = {1,2,3,4,5,6,7,8,9};
    int matrix2[3][3] = {1,2,3,4,5,6,7,8,9};
    int i,j,k matrix3[3][3];

  for(i=0; i<3; i++)
 { for(j=0; j<3;j++)
  { matrix3[j][j] = 0;}
 }
   for(i=0; i<3; i++)
 { for(j=0; j<3; j++)
  { for(k=0; k<3; k++)
 { matrix3[i][j] += matrix1[i][k] * matrix2[k][j];
 }
 }
}
printf("Result Matrix :\n");
```

Radhe

```
printf("Result:\n");
 for (int i=0; i<3; i++){
    for (int j=0; j<3; j++) {
    for (int k=0; k<3; k++) {
        printf("%d", result[i][j][k]);
    }
    printf("\n");
}
  return 0;
}
```

Conclusion:- In this practical I learnt about traversing in array and develop and execute a 'C' program to traverse and display the element from given array.



Conclusion :- In this practical I learnt abod traversing in array and develop and execute a 'C' program to traverse and display the element from given array.

Page No.

Date

Practical No : 2

Aim :- Develop and execute a c program to search a particular data from the given array using linear search.

Theory :- In linear search the list is searched sequentially and the position is returned if the key element to be searched is available in the list, otherwise -1 is returned. The search in linear search starts at the beginning of an array and move to the end testing for a match at each item.

Illustrate :- Find the position of 29 using linear search from 1, 15, 29, 31, 43

pass 1 :-

1	15	29	31	43

compare element $29 \neq 1$, 0th position

Pass 2 :-

1	15	29	31	43

compare element $29 \neq 15$, 1st position

Pass 3 :-

1	15	29	31	43

compare elemet $29 = 29$, 2nd position
element found at 2nd position.

Page No.

Date

Program :

```
#include <stdio.h>
int linear search (int arr[], int n, int key)
{
  for (int i = 0; i < n; i++){
  if (arr[i] == key){
     return i;
  }
  }
   return -I;
}
int main (){
    int n;
    printf("Enter the number of elements in the array");
    scanf("%d", & n);
   int arr[n]
   printf("Enter %d elements :\n", n);
   for (int i = 0; i < n; i++){
     scanf("%d", & arr [i]);
   }
   int key;
   printf("Enter the element to search for :");
   Scanf("%d", & they key);
   int result = linear search (arr, n, key);
```

Teacher's Signature

Page No.

Date

```
if (result != -1) {
    printf ("Element %d found at index %d\n", key
    result);
}
else {
    printf ("element %d not found in the array\n",
            key);
}
return 0;
}
```

Conclusion:- We have successfully studied to execute a c program to search a particular data from the given array using linear search.

Teacher's Signature

Practical No :- 3

Page No.

Date

Aim :- Develop & and execute a c programm to search a particular data from the given array using binary search.

Theory :- Binary search a particular item by comparing the middle most item of the collection. If match occurs then index of item is returned. If middle item is greater than item then item is searched in sub-array to the right of the middle item otherwise item is search in sub array to the left of the middle item.

This process continues on sub array as well until the size of subarray reduces to zero. In Binary search array should be sorted.

Illustrate :- Find the location of 35 using binary search from 10, 14, 19, 26, 27, 31, 33, 35, 42, 44

Pass 1 :-

0	1	2	3	4	5	6	7	8	9
10	14	19	26	27	31	33	35	42	44
↓ low									↓ High

mid = (low + high)/2

Here it is (0+9)/2 = 4 (integer value of 4.5) so 4 is mid array.

Page No.

Date

	0	1	2	3	4	5	6	7	8	9
Pass 2 :	10	14	19	26	27	31	33	35	42	44
						↓ low				↓ High

We change our low to mid + 1 and find the new mid value again

low = mid + 1

mid = (low + high)/2

our new mid is 7 now. we compare the value stored at location 7 with our target value 35.

0	1	2	3	4	5	6	7	8	9
10	14	19	26	27	31	33	35	42	44

The value stored at location 7 is match.

We conclude that target value 35 is stored at location 7.

Program :

```
#include <stdio.h>
int binary search (int arr[], int n, int target)
{
    int left = 0;
    int right = n - 1;

    while (left <= right) {
    int mid = left + (right - left) / 2;

    if (arr [mid] == target) {
        return mid;
    }
    if (arr [mid] < target) {
        left = mid + 1;
    }
    else
    {
        right = mid - 1;
    }
    }
    return -1;
}
```

Page No.

Date

```
int main()
{
  int arr[] = {2,4,6,8,10,12,14};
  int n = size of (arr) /size of int;
  int target = 10;
  int result = binary search (arr, n, target);
  printf ("the element %d was found at %d", target,
          result);

  return 0;
}
```

Conclusion:-

We have succesfully studied to execute a c programm to search a particular data from the given array using binary search.

Teacher's Signature

Page No.

Date

Practical No:- 4

Aim:- Develop and execute a c program to sort an array using Bubble sort.

Theory:- Bubble sort is an algorithm which is used to sort N elements that are given in a memory for e.g. an array with N number of elements. Bubble sort compare all the element one by one and sort them based on their values. It is called bubble sort because with each iteration the smaller element in the list bubbles up towards the first place, just like a water bubble rises up to the water surface.

Example:- Sort the following array 5, 1, 12, -5, 16 using bubble sort.

Pass 1 :- | 5 | 1 | 12 | -5 | 16 | 5 > 1 swap

| 1 | 5 | 12 | -5 | 16 | 5 < 12, ok

| 1 | 5 | 12 | -5 | 16 | 12 > -5 swap.

| 1 | 5 | -5 | 12 | 16 | 12 < 16 ok

Page No.

Date

Pass 2 : | 1 | 5 | -5 | 12 | 16 | 1<5, ok

| 1 | 5 | -5 | 12 | 16 | 5 > -5 swap.

| 1 | -5 | 5 | 12 | 16 | 5<12, ok

Pass 3 : | 1 | -5 | 5 | 12 | 16 | 1 > -5, swap

| -5 | 1 | 5 | 12 | 16 | 1 < 5 ~~5 < 1~~, ok.

Pass 4 : | -5 | 1 | 5 | 12 | 16 | . -5 < 1 ok.

| -5 | 1 | 5 | 12 | 16 | sorted.

Program :

```
#include <stdio.h>
void print array (int * a, int n)
{
    for (int i = n; i < n ; i ++) {
    printf ("%d", a[i]);
}
    printf ("\n");
}
void bubble sort (int * a , int n) {
    int temp;
    for (int i = 0 ; i <= n ; i ++) {
    for (int j = 0 ; j < n-1 -i ; j ++) {.
```


Teacher's Signature

Page No.

Date

```
        if (a[j] > a[j+1]){
            temp = a[j];
            a[j] = a[j+1];
            a[j+1] = temp;
        }
    }
}
}
int main(){
    int a[] = {4,5,1,3,8};
    int n = 5;
    print array (a,n);
    bubble sort (a,n);
    print array (a,n);

    return 0;
}
```

Conclusion: We have succesfully studied to execute a c program to sort a array using bubble sort.

Page No.

Date

Practical No : 5

Aim : Develop and execute a C program to sort an array using selection sort.

Theory : Selection sorting is conceptually the most simplest sorting algorithm. This algorithm first finds the smallest element in the array and exchange it with the element in the first position, then find the second smallest element and exchange it with the element in the second position and continues in this way untill the entire array is sorted.

Illustration :

	29	10	14	37	13	29 > 10 swap.
Pass 1 :	10	29	14	37	13	29 > 13 swap
Pass 2 :	10	13	14	37	29	
Pass 3 :	10	13	14	37	29	37 > 29 swap.
Pass 4 :	10	13	14	29	37	sorted.

Teacher's Signature

(Stores G.P.N.)

Page No.

Date

Program :-

```
#include <stdio.h>
void print array (int *a, int n)
{
  for (int i=0; i<n; i++)
  {
    printf ("%d", a[i]);
  }
}
void selection sort (int *a, int n)
{
  int index min, temp;
printf ("Selection sort will running");
  for (int i=0; i<n-1; i++){
    index min = i;
  for (int j=i+1; j<n; j++){
    if (a[j] < a[indexmin]){
     indexmin = j;
    }
  }
    temp = a[i];
    a[i] = a[indexmin];
    a[indexmin] = temp;
  }
}
```

Teacher's Signature

Page No.

Date

```
int main()
{
  int a[] = { 2, 6, 7, 3, 9};
  int n = 5;
  printf array (a,n);
  selection sort (a,n);
  print array (a,n);
  return 0;
}
```

Conclusion: We have successfully studied to execute a C program to sort on array using selection sort.

Teacher's Signature

Page No.

Date

Practical No:- 6

Aim:- Develop and execute a c program to sort an array using insertion sort.

Theory:- It is a simple sorting algorithm which sorts the array by shifting elements one by one following are some of the important characteristics of insertion sort.

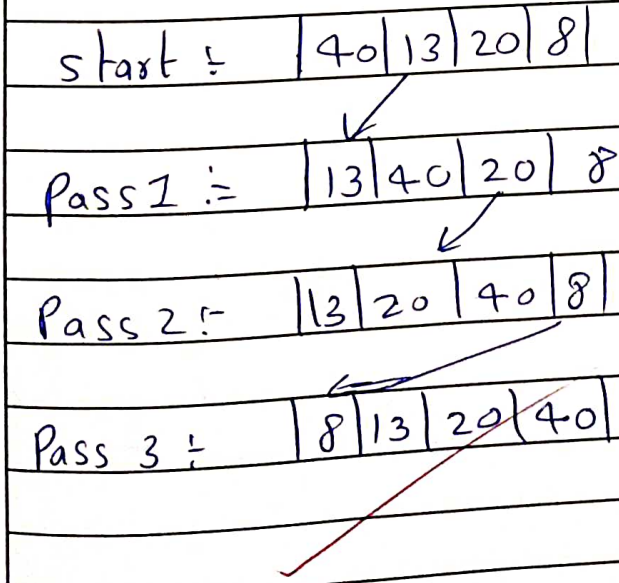
i) It has one of the simplest Implementation.

ii) It is efficient for smaller data sets, but ~~ev~~ very inefficient for larger lists.

iii) It is better than selection sort and bubble sort algorithms.

iv) It's space complexity is less.

Illustration:-



Teacher's Signature

Page No.
Date

Program:-

```
#include<stdio.h>
void print array (int* A, int n)
{
  for (int i=0; i<n; i++){
    printf ("%d", a[i]);
  }
  printf ("\n");
}
void insertion sort (int* A, int n)
{
 int key, j;
 for (int i=1; i<n; i++){
   key = A[i];
   j = i-1;

  while (j>=0 && A[j] >key) {
     A[j+1] = A[j];
     j--;
  }
   A[j+1] = key;
 }
}
```

Teacher's Signature

Page No.

Date

```
int main()
{
    int A[] = {2, 1, 3, 8, 12};
    int n = 5;
    print array (A, n);
    insertion sort (A, n);
    print array (A, n);
    return 0;
}
```

Conclusion:- We have succesfully studied to execute a c program to sort an array using insertion sort.

Teacher's Signature

Page No.

Date

Practical No:- 7

Aim:- Develop and execute a c program to perform push and pop operation based on stack using array.

Theory:- Stack is mechanism in which a number of elements are arranged one above other stack works units concpt of the LIFO (Last in first out). the elements which enters first will come out at last.

Push operation:-

1] Push operation refers to inserting on element in stack.

2] If we try to insert on element when the stack is full, then overflow condition occur.
If (top == dim - 1)

3] If the stack has space then increase top by 1 to point next empty space.

4] It add element to newly stack where top is pointing.

Pop operation:-

1] If stack has no element means it is empty then display the underflow.

2] If the stack has element deque top by 1 then Push, Pop operation in stack.

Page No.

Date

```
#include <stdio.h>
#include <malloc.h>

int top, dim;
int * a;
void create stack (int n){
    dim = n;
  a = malloc (size of (int) * n);
    top = -1;
}
void push (int ele) {
      if (top == dim -1){
  printf ("stack overflow");
}
else{
    top ++;
    a[top] = ele;
}
int pop (){
 if (top == -1){
 printf("Stack underflow");
    return 0;-1;
}
else{
    return a[top --];
}
```

Page No.

Date

```
void print (){
    if (top == -1){
    printf (" stack is empty");
  }
printf (" stack elements");
  for (int i = 0; i <= top; i++){
      printf ("%d", a[i]);
    }
    printf ("\n");
  }
int main (){
   int data;
   create stack[5];
    push(30);
    push(40);
    push (50);
    push (60);
    push (70);
   data = pop ();
    if (data == -1){
     printf ("Popped elements %d \n", data);
   {
    printf();
      return 0;
  }
```

Teacher's Signature

Page No.

Date

Conclusion:- Thus I successfully performed the push and pop operation in the stack.

Teacher's Signature

Page No.

Date

Practical No:9

Aim:- Develop and execute a c program to perform INSERT and DELETE operations on Queue using the array.

Theory:- A Queue is a linear data structure in which the order of operations FIFO (First In First out). The array in a data structure that contains memory location. In Queue the insertion and deletion operations are done of opposite end of the Queue.

Program:-

```
#include <stdio.h>
#include <malloc.h>
struct Queue{
    int * arr;
    int size;
    int front, rea;
};
void create (struct Queue * q, int n)
{
    q -> front = q -> rear = -1;
    q -> size = n;
    q -> arr = (int *) malloc (size of (int));
}
```

Page No.

Date

```
void enqueue(struct queue * q, int ele){
    if (q -> rear == q -> size-1){
    printf("\n queue overflow \n");
    return 0;
    }
    ele {
    q -> rear ++;
    q -> arr[q -> rear] = ele;}
}
void dequeue(struct queue * q){
    if (q -> rear == q -> front){
    printf("\nqueue underflow \n");
    q -> rear = q -> front = -1);
    return 0;
    }
else{
    q -> front ++;
    int ele = q -> arr[q -> front];
    printf("%d", ele);
}
}
int main (){
    struct queue a;
    int size = 10;
    create (&a, size);
```

Page No.

Date

```
for (int a=0; a<10; a++){
        enqueue (&a, a * 10);
}
For (int a=0; a<10; a++){
        dequeue (&a);
}
}
```

Conclusion: In this practical we learnt how to add elements in queue and how to delete using array.

Teacher's Signature

Page No.

Date

Practical No:-10

Aim:- Develop and execute a program for inserting deleting, traversing, searching, anode in circular singly ~~list~~ linked list

Theory :-

Programs :-

```
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node * next;
};
node * creat_node(){
    node * head;
    node * p;
    int n;
printf("enter size of linked list:");
scanf("%d",&n);
for(int k=0;k<n;k++){
    if(k==0){
head=(node *) malloc(size of (node));

p=head;
}
else{
p->next=(node *) malloc(size of
                        (node *));
p=p->next;
```

Teacher's Signature

```
}
printf("enter data of %d node:", k);
scanf("%d", &p->data);
}
p->next = Null;
return head;
}
void traversal (node * ptr){
    while (ptr != Null) {
    printf("element is %d \n", ptr->d); int;
     ptr = ptr-> next;
    }
}
node* insertion (node * prenode int value){
    node* ptr = (node *) malloc (sizeof(node));
     prenode -> next = ptr;
      ptr-> data = value;
}

Void search (node * ptr, int value) {
      while (ptr -> next != Null) {
        if (ptr -> data == value){
printf("element found \n");
  return 0;
}
```

Teacher's Signature

```
else{
    ptr = ptr -> next;
  }
}
if (ptr -> data == value) {
      printf ("element found \n");
  }
else{
    printf("element not found ! \n");
}

void delete (node * ptr, int index){
  int a = 0;
  node * p = ptr;
  node * q = ptr -> next;
    while (q < index -1){
      p = q -> next;
      q = p -> next;
      a++;
  }
  p -> next = q * -> next;
    free (q); }
int main (){
  node * head;
  head = create_node();
  printf ("\n\n");
```

Page No.

Date

```
traversal (head);
insertion (head (n));
printf ("\n");
traversal (head);
printf ("\n ptr deletion");
delete (head, 2);
traversal (head);
return 0;
}
```

Conclusion:- In this practical learnt about the how to use instead in our practical

Teacher's Signature

Page No.
Date

Practical No : 11

Aim :- Develop and execute a C program for inserting, Deleteing, traversing, searching a node in the circular linked list.

Theory :- Circlar linked list is a variation of a linked list where all the nodes are connected forming a node. this means that there is a no null at the end.

Program :-

```
#include <stdio.h>
#include <stdlib.h>

struct node{
  int data;
  struct node * next;
};
struct node *rear = Null;
void insert (int num){
  struct node * new node=(struct node *)
                malloc (size of(struct node));
newnode -> data = num;
  if (rear = = Null){
  rear = new node;
  rear -> next = rear;
}
```

Teacher's Signature

```
else{
    newnode -> next = rear -> next;
     rear -> next = new node;
}
void delete (){
    if (rear == Null){
    printf("empty stack underflow");
     return 0;
}
struct node * temp = rear -> next;
    if (rear -> next) == = rear) {
            free(rear);
           rear = Null;
     }
else{
    rear -> next = temp -> next;
     free (temp);
}
void display (){
    if (rear = = Null){
     printf ("stack overflow");
       return 0;
    }
    struct node * temp = rear -> next;
    printf ("\n circular linked list elements :
                          \n");
```

```
do{
printf("Node data: %d\n", temp->data);
    temp = temp -> next;
  }
while (temp != rear -> next);}
  void search (int num){
      int position = 0;
       if (rear = Null){
      printf ("\n Linked underflow");
       return;
      }

struct node * temp = rear -> next;
 do{
   position ~~it~~ ++;
    if (temp -> data == num){
printf ("\n element %d found at position
        %d in the linked list".num,position);
    return 0;
  }
temp = temp -> next;
}
while (temp != rear -> next){
  printf ("\n element %d not found in linked
        list", num);}
```

Page No.
Date

```
int main(){
    int choice, num, n;
    while (1){
    printf("circular linked list operator \n");
    printf("1: insert \n 2. Delete\n 3. Traverse \n
            4 - search\n 5. exit");
    printf("Enter your choice");
    scanf("%d", &choice);
     switch (choice)
{
    case 1; printf("enter no of element to
                    be inserted");
            scanf("%d", &n);
            printf("enter elements to insert:");
            for (int i=0; i<n; i++){
              scanf("%d", &num);
              insert (num); }
            break;

    case 2; delete();
            break;
    case 3: display();
            break;
    case 4; printf("enter element to search:");
            scanf("%d", &num);
            search (num);
            break;
```

Teacher's Signature

Page No.

Date

```
case 5 : exit (0); }

    default : { print
        printf("Invalid choice \n");
    }
}
return 0;
}
```

Conclusion :- In this practical I learnt about the circular linked list and executed the program succesfully.

Teacher's Signature

Page No.

Date

Practical No :- 12

Aim :- Develop and execute a program for inserting and deleting a node from binary tree.

Theory :-

In a binary tree, you can insert a node by haversing the tree and finding the appropriate location based on the node's value, creating a new node at that positions. If the value alredy exists, you can decide wheather to handle duplicates or not.

program :-

```
#include <stdio.h>
#include <stdlib.h>
 struct Node{
      int data;
      struct Node * left;
      struct Node * right;
 };
 struct Node * createNode( int data){
   struct Node * newNode = (struct Node *)
                malloc(sizeof(struct Node));
   newNode -> data = data;
   newNode -> left = Null;
   newNode -> right = Null;
    return newNode;
 }
```

Page No.

Date

```
struct Node * insert (struct Node * root, int dada){
    if (root == Null){
        return createNode (data);
    }
    if (data < root -> data){
        root -> left = insert (root -> left data);
    }
    else if (data > root -> data){
        root -> right = insert (root -> right, data);
    }
    return root;
}
struct Node * findmin (struct Node * root){
    while (root -> left -1 = Null){
        root = root -> left;
    } return root;
}
struct Node * delete (struct Node * root, int key){
    if (root == Null){
        return root;
    }
    if (key < root -> data){
        root -> left = delete (root -> left, key);
    } else if (key > root -> data){
        root -> right = delete Node (root -> right)
    }
```

Teacher's Signature

```
struct Node temp = findmin(root -> right);
    root -> data = temp -> data;
    root -> right = deleteNode (root -> right, temp -> data)
}
return root;
}
void inorder_Traversal (struct Node root){
    if (root == Null){
     return;
    }
    inorder Traversal (root -> left);
    printf("%d", root -> data);
     inorder Traversal (root -> right);
}
int main(){
     struct Node * root = Null;
    root = insert (root, 50);
    root = insert (root, 30);
    root = insert (root, 20);
    root = insert (root, 40);
    root = insert (root, 70);
    root = insert (root, 60);
    root = insert (root, 80);

printf("Inorder traversal before deletion");
    inorder Traversal (root);
    printf("\n")
```

Page No.

Date

```
root = deleteNode(root, 20);
printf("Inorder traversal after deletion: ");
inorderTraversal(root);
printf("\n");

return 0;
}
```

Conclusion: Through exicuting this practical we succesfully exicuted inserting, deleting operations in binary tree.

Teacher's Signature

Page No.

Date

Practical No:13

Aim: Develop and execute a program to traverse the tree in (inorder, preorder, postorder)

program:

```
#include<stdio.h>
#inclule<malloc.h>
c
struct node{
    struct node * left;
    struct node * right;
    int data;
};
node * create node (int value){
node * n = (node *) malloc (size of (node));
n->left = null;
n->right = Null;
n->data = value;
}
void tree traversal_in order (node * ptr){
    if (root != Null){
    tree traversal_inorder(root->left);
    Printf("%d \n", rool->data);
    tree traversal inorder (root->right);
}
void tree traversal_preorder (node * root){
    if (rod != Null){
    printf("%d /n", root->data);
```

Teacher's Signature

```
tree_traversal_preorder (~~root~~)(root->left);
tree_traversal_preorder (root->right);
}
}
void tree traversal_postorder (node * root) {
    if (root != Null)
    {
    tree traversal_postorder (node * root) {
        if (root != Null) {
    tree traversal postorder (root->left);
    tree traversal postorder (root->right);
    printf ("%d \t", root->data);
    }
    }
int main () {
node * n1 = create node (1);
node * n2 = create node (2);
node * n3 = create node (3);
node * n4 = create node (4);
node * n5 = create node (5);
node * n6 = cocate node (6);
node * n7 = create node (7);
node * n8 = create node (8);
node * n9 = create node (9);
node * n10 = create node (10);
node * n11 = create node (11);
```

Page No.

Date

```
n1 -> left = n2;
n1 -> right = n6;
n2 -> left = n3;
n2 -> right = n4;
n4 -> right = n5;
n6 -> right = n7;
n7 -> left = n8;
n8 -> left = n9;
n9 -> left = n10;
n9 -> right = n11;

printf ("Inorder: \n");
  tree traversal_inorder (n);

printf("\n Preorder : \n");
  tree traversal_preorder (n);

printf ("\n postorder : \n");
  tree traversal_postorder (n);
  return 0;
}
```

Conclusion : In this practical we learnt about tree and how to traverse in order, preorder and in postorder form.

(Students Co- op Stores G.P.N.)

Teacher's Signature