Community Institute OF management Studies
MCA I semester
LAB MANUAL

1. Given {4,7,3,2,1,7,9,0} find the location of 7 using Linear and Binary search and also display its first occurrence

```
# include<stdio.h>
# include<conio.h>
void LINEAR_SEARCH(int a[10], int n, int key)
{
int i,found=0;
for(i=0;i<n;i++)
{
if(key==a[i])
{
printf("\n %d found at a[%d]",key,i);
found=1;
break;
}
}
if(found==0)
printf("\n Element not found in the list");
}
void BINARY_SEARCH(int a[10], int key, int first, int last)
{
int mid;
mid=(first+last)/2;
if(key < a[mid])
BINARY_SEARCH(a,key,first,mid-1);
else if(key > a[mid])
BINARY_SEARCH(a,key,mid+1,last);
else
printf("\n %d found at a[%d]",key,mid);
}
void DISPLAY_ARRAY(int a[10],int n)
```

```
{
int i;
printf("\n Given array : ");
for(i=0;i<n;i++)
{
printf("%3d",a[i]);
}
}
void main()
{
int a[8] = {4,7,3,2,1,7,9,0};
int sa[8] = {0,1,2,3,4,7,7,9};
int n=8;
int key=7;
int choice;
clrscr();
printf("\n 1. Linear Search");
printf("\n 2. Binary Search");
printf("\n Enter your choice :");
scanf("%d",&choice);
switch(choice)
{
case 1 : {
DISPLAY_ARRAY(a,n);
LINEAR_SEARCH(a,n,key);
break;
}
case 2 : {
DISPLAY_ARRAY(sa,n);
BINARY_SEARCH(sa,key,0,n-1);
break;
}
default : printf("\n Invalid choice ");
}
getch();}
```

OUTPUT :

```
LINEAR SEARCH :
1. Linear Search
2. Binary Search
 Enter your choice :1

 Given array :  4  7  3  2  1  7  9  0
 7 found at a[1]

BINARY SEARCH:

1. Linear Search
2. Binary Search
 Enter your choice :2

 Given array :  0  1  2  3  4  7  7  9
 7 found at a[5]
```

2. Program to perform quick sort in ascending order given {5, 3, 1, 6, 0 2 4}

```
#include<stdio.h>
void quicksort(int number[25],int first,int last){
  int i, j, pivot, temp;
  if(first<last){
    pivot=first;
    i=first;
    j=last;
    while(i<j){
      while(number[i]<=number[pivot]&&i<last)
      i++;
      while(number[j]>number[pivot])
      j--;
      if(i<j){
        temp=number[i];
        number[i]=number[j];
        number[j]=temp;
      }
    }
```

```
    temp=number[pivot];
    number[pivot]=number[j];
    number[j]=temp;
    quicksort(number,first,j-1);
    quicksort(number,j+1,last);
  }
}
int main(){
  int i, count, number[25];
  printf("How many elements are u going to enter?: ");
  scanf("%d",&count);
  printf("Enter %d elements: ", count);
  for(i=0;i<count;i++)
  scanf("%d",&number[i]);
  quicksort(number,0,count-1);
  printf("Order of Sorted elements: ");
  for(i=0;i<count;i++)
  printf(" %d",number[i]);
  return 0;
}
```

Output:

How many elements are u going to enter?: 7
Enter 7 elements: 5 8 2 1 10 4 6
Order of Sorted elements: 1 2 4 5 6 8 10

3. Perform the merge sort on the input {75,8,1,16,48,3,7,0} display the output in descending order

```
#include <stdio.h>
int main() {
  int size1, size2, size3;
  printf("\nEnter the size for the first array: ");
  scanf("%d", & size1);
  printf("\nEnter the size for the second array: ");
```

```
scanf("%d", & size2);
size3 = size1 + size2;
printf("\nEnter the elements :");
 /*Array Declaration*/
int array1[size1], array2[size2], array3[size3];
/*Array Initialized*/
for (int i = 0; i < size1; i++) {
 scanf("%d", & array1[i]);
 array3[i] = array1[i];
}
int k = size1;
printf("\nEnter the elements:");
/*Array Initialized*/
for (int i = 0; i < size2; i++) {
 scanf("%d", & array2[i]);
 array3[k] = array2[i];
 k++;
}
printf("\nmerged array of first and second:\n");
for (int i = 0; i < size3; i++)
 /*Printing the merged array*/
 printf("%d ", array3[i]);
printf("\nsorted array in descending order\n");
/*Sorting Array*/
for (int i = 0; i < size3; i++) {
 int temp;
 for (int j = i + 1; j < size3; j++) {
  if (array3[i] < array3[j]) {
   temp = array3[i];
   array3[i] = array3[j];
   array3[j] = temp;
  }
 }
}
/*Printing the sorted Array*/
for (int i = 0; i < size3; i++) {
 printf(" %d ", array3[i]);
```

```
    }
    return 0;
  }
```

Output:

Enter the size for the first array: 8
Enter the size for the second array: 7
Enter the elements :2 4 6 9 1 0 7 3
Enter the elements:10 9 13 12 5 8 15
merged array of first and second:
2 4 6 9 1 0 7 3 10 9 13 12 5 8 15
sorted array in descending order
15 13 12 10 9 9 8 7 6 5 4 3 2 1 0

4. **Write a program to insert the elements {61,16,8,27} into singly linked list and delete 8,61,27 from the list. Display your list after each insertion and deletion.**

```
// C program to insert {61,16,8,27} and delete {8,61,27} from the list
# include<stdio.h>
# include<conio.h>
# include<alloc.h>
# include<ctype.h>
typedef struct node
{
int info;
struct node *link;
}NODE;
NODE *header=NULL;
void DISPLAY()
{
NODE *start=header;
printf("\n *** LIST *** : ");
while(start!=NULL)
{
printf("%4d",start->info);
start=start->link;
```

```
}
}
void INSERT(int item)
{
NODE *newnode,*curptr;
newnode = (NODE *) malloc(sizeof(NODE));
newnode->info=item;
newnode->link=NULL;
if(header==NULL)
header=newnode;
else
{
curptr=header;
while(curptr->link !=NULL)
{
curptr=curptr->link;
}
curptr->link=newnode;
}
DISPLAY();
}
void DELETE(int item)
{
NODE *curptr=header, *prevptr=header;
if(header==NULL)
{
printf("\n EMPTY LIST");
}
else if(header->info==item)
{
header=header->link;
free(curptr);
}
else
{
while(curptr!=NULL)
{
if(curptr->info==item)
{
```

```
prevptr->link=curptr->link;
free(curptr);
curptr=curptr->link->link;
}
else
{
prevptr=curptr;
curptr=curptr->link;
}
}
}
DISPLAY();
}
void main()
{
int item,choice;
clrscr();
printf("\n Insertion :");
INSERT(61);
INSERT(16);
INSERT(8);
INSERT(27);
printf("\n Deletion :");
DELETE(8);
DELETE(61);
DELETE(27);
getch();
}
```

OUTPUT:

Insertion :

***** LIST *** : 61**

***** LIST *** : 61 16**

***** LIST *** : 61 16 8**

***** LIST *** : 61 16 8 27**

Deletion :

***** LIST *** : 61 16 27**

***** LIST *** : 16 27**

***** LIST *** : 16**

5. Write a program to add 6x3+10x2+0x+5 and 4x2+2x+1 using linked list.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
struct polynomial
{
int coeff;
int power;
struct polynomial *LINK;
};
typedef struct polynomial NODE;
NODE *poly1=NULL,*poly2=NULL,*poly3 = NULL;
NODE *create_poly();
NODE *add_poly(NODE *poly1,NODE *poly2);
void display_poly(NODE *ptr);
/* To create the polynomial*/
NODE *create_poly()
{
int flag;
int coeff,pow;
NODE *tmp_node =(NODE *)malloc(sizeof(NODE));//create thefirst
node
NODE *poly=tmp_node;
do
{
printf("\n Enter coeff:");
scanf("%d",&coeff);
tmp_node->coeff=coeff;
printf("\n Enter Pow:");
scanf("%d",&pow);
tmp_node->power = pow;
tmp_node->LINK=NULL;
printf("\n Do you want to add more terms? (Y=1/N=0):");
scanf("%d",&flag);
if(flag==1)
{
tmp_node->LINK=(NODE *) malloc(sizeof(NODE));
tmp_node = tmp_node->LINK;
```

```
tmp_node -> LINK = NULL;
}
} while(flag);
return poly;
}
/*add two polynomial */
NODE *add_poly(NODE *poly1, NODE *poly2)
{
NODE *tmp_node,*poly;//Temporary storage for the linked list
tmp_node=(NODE *)malloc(sizeof(NODE));
tmp_node->LINK = NULL;
poly3=tmp_node;
//Loop while both of the linked list have value
while(poly1&&poly2)
{
if(poly1->power > poly2->power)
{
tmp_node->power=poly1->power;
tmp_node->coeff=poly1->coeff;
poly1=poly1->LINK;
}
else if (poly1->power < poly2->power)
{
tmp_node->power = poly2-> power;
tmp_node->coeff =poly2->coeff;
poly2 = poly2->LINK;
}
else
{
tmp_node->power = poly1->power;
tmp_node->coeff = poly1->coeff+poly2->coeff;
poly1=poly1->LINK;
poly2=poly2->LINK;
}
if(poly1&&poly2)
{
tmp_node->LINK=(NODE *)malloc(sizeof(NODE));
tmp_node=tmp_node->LINK;
tmp_node->LINK=NULL;
}
```

```
}
//Loop while either of the linked list has value
while(poly1||poly2)
{
tmp_node->LINK =(NODE *)malloc(sizeof(NODE));
tmp_node=tmp_node->LINK;
tmp_node->LINK=NULL;
if(poly1)
{
tmp_node->power=poly1->power;
tmp_node->coeff=poly1->coeff;
poly1=poly1->LINK;
}
if(poly2)
{
tmp_node->power=poly2->power;
tmp_node->coeff=poly2->coeff;
poly2=poly2->LINK;
}
}
}
/* Display polynomial */
void display(NODE *ptr)
{
while(ptr!=NULL)
{
printf("%dX^%d",ptr->coeff,ptr->power);
ptr=ptr->LINK;
if(ptr!=NULL)
printf(" + ");
}
}
int main()
{
clrscr();
printf("\n Create 1st Polynomial: ");
poly1=create_poly();
printf("\n First polynomial : ");
display(poly1);
printf("\n Create 2nd Polynomial:");
```

```
poly2=create_poly();
printf("\n Second polynomial :");
display(poly2);
add_poly(poly1,poly2);
printf("\n Addition of Two polynomials : ");
display(poly3);
getch();
}
```

OUTPUT:

Create 1st Polynomial:
Enter coeff:6
Enter Pow:3
Do you want to add more terms? (Y=1/N=0):1
Enter coeff:10
Enter Pow:2
Do you want to add more terms? (Y=1/N=0):1
Enter coeff:0
Enter Pow:1
Do you want to add more terms? (Y=1/N=0):1
Enter coeff:5
Enter Pow:0
Do you want to add more terms? (Y=1/N=0):0
First polynomial : 6X^3 + 10X^2 + 0X^1 + 5X^0
Create 2nd Polynomial:
Enter coeff:4
Enter Pow:2
Do you want to add more terms? (Y=1/N=0):1
Enter coeff:2
Enter Pow:1
Do you want to add more terms? (Y=1/N=0):1
Enter coeff:1
Enter Pow:0
Do you want to add more terms? (Y=1/N=0):0
Second polynomial :4X^2 + 2X^1 + 1X^0
Addition of Two polynomials : 6X^3 + 14X^2 + 2X^1 + 6X^0

6. Write a program to push 5,9,34,17,32 into stack and pop 3 times from the stack, also display the popped numbers

```
# include<stdio.h>
# include<conio.h>
# define MAX 5
int STACK[MAX];
int TOP=-1;
void DISPLAY();
void PUSH(int item)
{
if(TOP==MAX-1)
{
printf("\n STACK Overflow");
}
else
{
printf("\n ** PUSH %d **",item);
TOP=TOP+1;
STACK[TOP]=item;
DISPLAY();
}
}
void POP()
{
if(TOP==-1)
{
printf("\n STACK Underflow");
getch();
}
else
{
printf("\n ** %d POPED **",STACK[TOP]);
TOP=TOP-1;
DISPLAY();
}
}
void DISPLAY()
{
int i;
```

```
for(i=TOP;i>=0;i--)
{
printf("\n STACK[%d]=%d",i,STACK[i]);
}
getch();
}
void main()
{
int i;
clrscr();
printf("\n PUSH 5,9,34,17,32\n");
PUSH(5);
PUSH(9);
PUSH(34);
PUSH(17);
PUSH(32);
printf("\n POP 3 elements\n");
POP();
POP();
POP();
}
```

OUTPUT

PUSH 5,9,34,17,32

**** PUSH 5 ****
STACK[0]=5
**** PUSH 9 ****
STACK[1]=9
STACK[0]=5
**** PUSH 34 ****
STACK[2]=34
STACK[1]=9
STACK[0]=5
**** PUSH 17 **STACK[3]=17**
STACK[2]=34
STACK[1]=9
STACK[0]=5
**** PUSH 32 ****

```
STACK[4]=32
STACK[3]=17
STACK[2]=34
STACK[1]=9
STACK[0]=5
POP 3 elements

** 32 POPED **
STACK[3]=17
STACK[2]=34
STACK[1]=9
STACK[0]=5
** 17 POPED **
STACK[2]=34
STACK[1]=9
STACK[0]=5
** 34 POPED **
STACK[1]=9
STACK[0]=5
```

7. Write a recursive program to find GCD of 4,6,8.

```
# include<stdio.h>
# include<conio.h>
int GCD(int m, int n)
{
if(n==0)
{
return(m);
}
else if(n>m)
{
return(GCD(n,m));
}
else
{
return(GCD(n,m%n));
}
}
```

```
void main()
{
int gcd12, gcd3;
clrscr();
gcd12=GCD(4,6);
printf("\n GCD between 4 & 6 = %d",gcd12);
gcd3=(GCD(gcd12,8));
printf("\n GCD between 4,6 & 8 = %d",gcd3);
getch();
}
```

OUTPUT:

GCD between 4 & 6 = 2

GCD between 4,6 & 8 = 2

8. Write a program to insert the elements {5,7,0,6,3,9} into Circular queue and delete three elements from the list. Display your list after each insertion and deletion.

```
#include <stdio.h>
#define size 6
int front = - 1;
int rear = - 1;
int queue[size];
void display_CQ()
{
int i;
printf("\n Circular Queue : ");
if (front > rear)
{
for (i = front; i < size; i++)
{
printf("%d ->", queue[i]);
}
for (i = 0; i <= rear; i++)
printf("%d -> ", queue[i]);
}
```

```
else
{
for (i = front; i <= rear; i++)
printf("%d ->", queue[i]);
}
printf("[%d]",queue[front]);
getch();
}
void insert_CQ(int item)
{
if ((front == 0 && rear == size - 1) || (front == rear + 1))
{
printf("queue is full");
return;
}
else if (rear == - 1)
{
rear++;
front++;
}
else if (rear == size - 1 && front > 0)
{
rear = 0;
}
else
{
rear++;
}
queue[rear] = item;
display_CQ();
}
void delete_CQ()
{
if (front == - 1)
{
printf("Queue is empty ");
}
```

```
else if (front == rear)
{
front = - 1;
rear = - 1;
}
else
{
front++;
}
display_CQ();
}
int main()
{
clrscr();
printf("\n *** Insertion *** : ");
insert_CQ(5);
insert_CQ(7);
insert_CQ(0);
insert_CQ(6);
insert_CQ(3);
insert_CQ(9);
printf("\n *** Deletion *** : ");
delete_CQ();
delete_CQ();
delete_CQ();
delete_CQ();
}
```

OUTPUT: * Insertion *** :**
Circular Queue : 5 ->[5]
Circular Queue : 5 ->7 ->[5]
Circular Queue : 5 ->7 ->0 ->[5]
Circular Queue : 5 ->7 ->0 ->6 ->[5]
Circular Queue : 5 ->7 ->0 ->6 ->3 ->[5]
Circular Queue : 5 ->7 ->0 ->6 ->3 ->9 ->[5]
***** Deletion *** :**
Circular Queue : 7 ->0 ->6 ->3 ->9 ->[7]
Circular Queue : 0 ->6 ->3 ->9 ->[0]

Circular Queue : 6 ->3 ->9 ->[6]
Circular Queue : 3 ->9 ->[3]

9. Given S1={"Flowers"} ; S2={"are beautiful"} I. Find the length of S1 II. Concatenate S1 and S2 III. Extract the substring "low" from S1 IV. Find "are" in S2 and replace it with "is"

```
# include<stdio.h>
# include<conio.h>
# include<string.h>
int LENGTH(char *str)
{
int i=0, len=0;
while(str[i]!='\0')
{
len++;
i++;
}
return(len);
}
void CONCAT(char *str1, char *str2)
{
int i=0,j=0;
while(str1[i]!='\0')
{
i++;
}
while(str2[j]!='\0')
{
str1[i]=str2[j];
i++;
j++;
}
str1[i]='\0';
printf("\n Concatenated string = %s",str1);
}
void EXTRACT(char *str,int pos, int elen)
{
int i=0,j=0;
char substr[10];
```

```
for(i=pos;i<=elen;i++)
{
substr[j]=str[i];
j++;
}
substr[j]='\0';
printf("\n Substring = %s",substr);
}
void REPLACE(char *str, char *sstr, char *rstr, int pos)
{
char output[50];
int i=0,j=0,k=0;
for(i=0;i<LENGTH(str);i++)
{
if(i==pos)
{
for(k=pos;k<LENGTH(rstr);k++)
{
output[j]=rstr[k];
j++;
i++;
}
}
else
{
output[j]=str[i];
j++;
}
}
output[j]='\0';
printf("\n Output = %s",output);
getch();
}
void main()
{
char *S1,*S2;
int len,choice,pos,elen;
while(1)
{
clrscr();
```

```
strcpy(S1,"Flowers");
strcpy(S2,"are beautiful");
printf("\n S1 = %s S2 = %s",S1,S2);
printf("\n 1. Length 2.Concatenate 3.Extract Substring 4.REPLACE
5.Exit\n");
printf("\n Enter your choice : ");
scanf("%d",&choice);
switch(choice)
{
case 1 : {
len = LENGTH(S1);
printf("\n Length of %s = %d",S1,len);
}break;
case 2: {
CONCAT(S1,S2);
}break;
case 3: { printf("\n Enter position & length of substring in S1 : ");
scanf("%d %d",&pos,&elen);
EXTRACT(S1,pos,elen);
}break;
case 4: {
REPLACE(S2,"are","is",0);
}break;
case 5: exit(0);
default : printf("\n Invalid option");
}
getch();
}
}
```

OUTPUT:

S1 = Flowers S2 = are beautiful

1. Length 2. Concatenate 3. Extract Substring 4. Replace 5. Exit

Enter your Choice: 1

Length of Flowers = 7

S1 = Flowers S2 = are beautiful

1. Length 2. Concatenate 3. Extract Substring 4. Replace 5. Exit

Enter your Choice: 2
Concatenated of String: Flowersare beautiful

S1 = Flowers S2 = are beautiful
1. Length 2. Concatenate 3. Extract Substring 4. Replace 5. Exit

Enter your Choice: 3
Enter position & length of substring: 1 3
Substring = low

S1 = Flowers S2 = are beautiful
1. Length 2. Concatenate 3. Extract Substring 4. Replace 5. Exit

Enter your Choice: 4
Output = is beautiful

10. Write a program to convert an infix expression x^y/(5*z)+2 to its postfix expression

```
#include <stdio.h>
#include <ctype.h>
#define SIZE 50
char stack[SIZE];
int top=-1;
push(char elem)
{
stack[++top]=elem;
}
char pop()
{
return(stack[top--]);
}
int pr(char symbol)
{
if(symbol == '^')
{
return(3);
}
else if(symbol == '*' || symbol == '/')
```

```
{
return(2);
}
else if(symbol == '+' || symbol == '-')
{
return(1);
}
else
{
return(0);
}
}
void main()
{
char infix[50],postfix[50],ch,elem;
int i=0,k=0;
clrscr();
printf("Enter Infix Expression : ");
scanf("%s",infix);
push('#');
while( (ch=infix[i++]) != '\0')
{
if( ch == '(') push(ch);
else
if(isalnum(ch)) postfix[k++]=ch;
else
if( ch == ')')
{
while( stack[top] != '(')
postfix[k++]=pop();
elem=pop();
}
else
{
while( pr(stack[top]) >= pr(ch) )
postfix[k++]=pop();
push(ch);
}
}
while( stack[top] != '#')
```

```
postfix[k++]=pop();
postfix[k]='\0';
printf("\nPostfix Expression = %s\n",postfix);
getch(); }
```

OUTPUT:

Enter Infix Expression : x^y/(5*z)+2

Postfix Expression = xy^5z*/2+

11. Write a program to evaluate a postfix expression 5 3+8 2 - *.

```
#include<stdio.h>
int stack[20];
int top = -1;
void push(int x)
{
   stack[++top] = x;
}

int pop()
{
   return stack[top--];
}
int main()
{
   char exp[20];
   char *e;
   int n1,n2,n3,num;
   printf("Enter the expression :: ");
   scanf("%s",exp);
   e = exp;
   while(*e != '\0')
   {
      if(isdigit(*e))
      {
         num = *e - 48;
         push(num);
      }
```

```
        else
        {
           n1 = pop();
           n2 = pop();
           switch(*e)
           {
           case '+':
           {
              n3 = n1 + n2;
              break;
           }
           case '-':
           {
              n3 = n2 - n1;
              break;
           }
           case '*':
           {
              n3 = n1 * n2;
              break;
           }
           case '/':
           {
              n3 = n2 / n1;
              break;
           }
           }
           push(n3);
        }
        e++;
     }
     printf("\nThe result of expression %s  =  %d\n\n",exp,pop());
     return 0;
  }
```

OUTPUT:
Enter the expression :: 53+82-*
The result of expression 53+82-* = 48

12. Write a program to create a binary tree with elements 18,15,40,50,30,17,41 after creation insert 45 & 19 into tree & delete 15, 17 & 41 from tree. Display the tree on each insertion & deletion operation.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
struct node{
   int data;
   struct node *left;
   struct node *right;
};
 struct node *root= NULL;
 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;
}

//insert() will add new node to the binary search tree
void insert(int data) {
   //Create a new node
   struct node *newNode = createNode(data);
    if(root == NULL){
      root = newNode;
      return;
    }
   else {
       struct node *current = root, *parent = NULL;
           while(true) {
```

```
            parent = current;
             if(data < current->data) {
            current = current->left;
            if(current == NULL) {
               parent->left = newNode;
               return;
            }
          }
           else {
            current = current->right;
            if(current == NULL) {
               parent->right = newNode;
               return;
            }
          }
       }
    }
}
struct node* minNode(struct node *root) {
   if (root->left != NULL)
      return minNode(root->left);
   else
      return root;
}
struct node* deleteNode(struct node *node, int value) {
   if(node == NULL){
      return NULL;
    }
   else {
      if(value < node->data)
        node->left = deleteNode(node->left, value);
      else if(value > node->data)
        node->right = deleteNode(node->right, value);
      else {
         if(node->left == NULL && node->right == NULL)
          node = NULL;
```

```
            else if(node->left == NULL) {
               node = node->right;
            }
            else if(node->right == NULL) {
               node = node->left;
            }
            else {
                    struct node *temp = minNode(node->right);
                node->data = temp->data;
                 node->right = deleteNode(node->right, temp->data);
            }
         }
         return node;
      }
}
void inorderTraversal(struct node *node) {
    if(root == NULL){
       printf("Tree is empty\n");
         return;
    }
    else {
            if(node->left!= NULL)
          inorderTraversal(node->left);
       printf("%d ", node->data);
       if(node->right!= NULL)
         inorderTraversal(node->right);
        }
}
 int main()
{
   insert(18);
   insert(15);
   insert(50);
   insert(30);
   insert(17);
   insert(41);
```

```
    insert(45);
    insert(19);
    printf("Binary search tree after insertion: \n");
      inorderTraversal(root);
       struct node *deletedNode = NULL;
       deletedNode = deleteNode(root, 15);
    printf("\nBinary search tree after deleting node 15: \n");
    inorderTraversal(root);
      deletedNode = deleteNode(root, 17);
    printf("\nBinary search tree after deleting node 17: \n");
    inorderTraversal(root);
     deletedNode = deleteNode(root, 41);
    printf("\nBinary search tree after deleting node 41: \n");
    inorderTraversal(root);
      return 0;
}
```

OUTPUT:
Binary search tree after insertion:
15 17 18 19 30 41 45 50
Binary search tree after deleting node 15:
17 18 19 30 41 45 50
Binary search tree after deleting node 17:
18 19 30 41 45 50
Binary search tree after deleting node 41:
18 19 30 45 50

13. Write a program to create binary search tree with the elements {2,5,1,3,9,0,6} and perform inorder, preorder and post order traversal.

```
# include <stdio.h>
# include <conio.h>
# include <stdlib.h>
typedef struct BST {
int data;
struct BST *lchild, *rchild;
```

```
} node;
node *create_node() {
node *temp;
temp = (node *) malloc(sizeof(node));
temp->lchild = NULL;
temp->rchild = NULL;
return temp;
}
void insert(node *root, node *new_node) {
if (new_node->data < root->data) {
if (root->lchild == NULL)
root->lchild = new_node;
else
insert(root->lchild, new_node);
}
if (new_node->data > root->data) {
if (root->rchild == NULL)
root->rchild = new_node;
else
insert(root->rchild, new_node);
}
}
void inorder(node *temp) {
if (temp != NULL) {
inorder(temp->lchild);
printf("%3d", temp->data);
inorder(temp->rchild);
}
}
void preorder(node *temp) {
if (temp != NULL) {
printf("%3d", temp->data);
preorder(temp->lchild);
preorder(temp->rchild);
}
}
void postorder(node *temp) {
if (temp != NULL)
{
postorder(temp->lchild);
```

```
postorder(temp->rchild);
printf("%3d", temp->data);
}
}
void main()
{
int n=7,i=1;
node *new_node, *root;
node *create_node();
root = NULL;
clrscr();
printf("\nProgram For Binary Search Tree ");
for(i=1;i<=n;i++)
{
new_node = create_node();
printf("\nEnter The Element ");
scanf("%d", &new_node->data);
if (root == NULL) /* Tree is not Created */
root = new_node;
else
insert(root, new_node);
}
printf("\nThe Inorder display : ");
inorder(root);
printf("\nThe Preorder display : ");
preorder(root);
printf("\nThe Postorder display : ");
postorder(root);
getch();
}
```

OUTPUT:

Program For Binary Search Tree
Enter The Element 2
Enter The Element 5
Enter The Element 1
Enter The Element 3
Enter The Element 9
Enter The Element 0
Enter The Element 6

The Inorder display : 0 1 2 3 5 6 9
The Preorder display : 2 1 0 5 3 9 6
The Postorder display : 0 1 3 6 9 5 2

14. Write a program to sort elements using heap sort in { 9,16,32,8,4,1,5,8,0).

```
#include <stdio.h>
void heapify(int a[], int n, int i)
{
   int largest = i; // Initialize largest as root
   int left = 2 * i + 1; // left child
   int right = 2 * i + 2; // right child
   if (left < n && a[left] > a[largest])
      largest = left;
   if (right < n && a[right] > a[largest])
      largest = right;
   if (largest != i) {
      int temp = a[i];
      a[i] = a[largest];
      a[largest] = temp;
      heapify(a, n, largest);
   }
}
void heapSort(int a[], int n)
{
   for (int i = n / 2 - 1; i >= 0; i--)
      heapify(a, n, i);
   for (int i = n - 1; i >= 0; i--) {
      int temp = a[0];
      a[0] = a[i];
      a[i] = temp;
       heapify(a, i, 0);
   }
}
void printArr(int arr[], int n)
{
```

```
    for (int i = 0; i < n; ++i)
    {
        printf("%d", arr[i]);
        printf(" ");
    }
}
int main()
{
    int a[] = {9,16,32,8,4,1,5,8,0};
    int n = sizeof(a) / sizeof(a[0]);
    printf("Before sorting array elements are - \n");
    printArr(a, n);
    heapSort(a, n);
    printf("\nAfter sorting array elements are - \n");
    printArr(a, n);
    return 0;
}
```

OUTPUT:
Before sorting array elements are -
9 16 32 8 4 1 5 8 0
After sorting array elements are -
0 1 4 5 8 8 9 16 32

15. Write a program to display the Fibonacci series

```
#include<stdio.h>
int fibo_num (int i)
{
// if the num i is equal to 0, return 0;
if ( i == 0)
{
return 0;
}
if ( i == 1)
{
return 1;
```

```
}
return fibo_num (i - 1) + fibo_num (i -2);
}
int main ()
{
int i;
for ( i = 0; i < 10; i++)
{
printf (" %d \t ", fibo_num (i));
}
return 0;
}
```

OUTPUT:

0 1 1 2 3 5 8 13 21 34

16. Write a program to display the odd & even numbers

```
#include <stdio.h>
# include<conio.h>
void odd(); // Add 1 when the function is odd()
void even(); // Subtract 1 when the function is even
int num = 1; // global variable
void odd ()
{
    if (num <= 10)
  {
     printf (" %d ", num + 1);  // print a number by adding 1
     num++; // increment by 1
     even(); // invoke the even function
  }
  return;
}
void even ()
{
    if ( num <= 10)
  {
     printf (" %d ", num - 1); // print a number by subtracting 1
```

```
        num++;
        odd(); // call the odd() function
    }
    return;
}
int main ()
{
    odd(); // main call the odd() function at once
    return 0;
}
```

OUTPUT:
2 1 4 3 6 5 8 7 10 9

17. Write a program to display the factorial of a number

```
#include <stdio.h>
unsigned long factorial(int n)
{
    // base case: if `n` is 0 or 1
    if (n < 1) {
        return 1;
    }
     // use the recurrence relation
    return n * factorial(n - 1);
}
 int main()
{
    int n = 5;
    printf("The Factorial of %d is %lu", n, factorial(n));
     return 0;
}
```

OUTPUT:
The Factorial of 5 is 120

18. Write a program to illustrate for Tower of Hanoi

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

```
void TOH(int n, char source, char destination, char helper_t)
{
   if(n==0){
      return 0;
   }
   TOH(n-1,source, helper_t,destination);
   printf("\n Move disc %d from tower %c to tower %c",n, source,
    destination);
   TOH(n-1, helper_t, destination,source);
}
int main()
{
   TOH(4, 'A','B','C');
   return 0;
}
```

OUTPUT:

Move disc 1 from tower A to tower C

Move disc 2 from tower A to tower B

Move disc 1 from tower C to tower B

Move disc 3 from tower A to tower C

Move disc 1 from tower B to tower A

Move disc 2 from tower B to tower C

Move disc 1 from tower A to tower C

Move disc 4 from tower A to tower B

Move disc 1 from tower C to tower B

Move disc 2 from tower C to tower A

Move disc 1 from tower B to tower A

Move disc 3 from tower C to tower B

Move disc 1 from tower A to tower C

Move disc 2 from tower A to tower B

Move disc 1 from tower C to tower B

19. Given {5,3,1,6,0,2,4} order the numbers in ascending order using Bubble Sort Algorithm

```
# include<stdio.h>
# include<conio.h>
void BUBBLE_SORT(int a[], int n)
{
int pass,temp,j,i;
for(pass=1;pass<=n-1;pass++)
{
for(j=0;j<=n-pass-1;j++)
{
if(a[j] > a[j+1])
{
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp; { }
printf("\n Array after %d pass --->",pass);
for(i=0;i<n;i++)
printf("%3d",a[i]); }
}
void main()
{
int a[7]={5,3,1,6,0,2,4};
int n=7;
clrscr();
printf("\n Input arrays : 5 3 1 6 0 2 4");
BUBBLE_SORT(a,n);
getch();
}
```

OUTPUT:

Input arrays : 5 3 1 6 0 2 4
Array after 1 pass ---> 3 1 5 0 2 4 6
Array after 2 pass ---> 1 3 0 2 4 5 6
Array after 3 pass ---> 1 0 2 3 4 5 6
Array after 4 pass ---> 0 1 2 3 4 5 6

```
Array after 5 pass ---> 0 1 2 3 4 5 6
Array after 6 pass ---> 0 1 2 3 4 5 6
```

20. Write a program for Reversal of String using stack

```
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct Stack {
   int top;
   unsigned capacity;
   char* array;
};
struct Stack* createStack(unsigned capacity)
{
   struct Stack* stack
      = (struct Stack*)malloc(sizeof(struct Stack));
   stack->capacity = capacity;
   stack->top = -1;
   stack->array
      = (char*)malloc(stack->capacity * sizeof(char));
   return stack;
}
int isFull(struct Stack* stack)
{
   return stack->top == stack->capacity - 1;
}
int isEmpty(struct Stack* stack)
{
   return stack->top == -1;
}
void push(struct Stack* stack, char item)
{
   if (isFull(stack))
      return;
   stack->array[++stack->top] = item;
}
char pop(struct Stack* stack)
{
```

```
    if (isEmpty(stack))
        return INT_MIN;
    return stack->array[stack->top--];
}
void reverse(char str[])
{
    int n = strlen(str);
    struct Stack* stack = createStack(n);
    int i;
    for (i = 0; i < n; i++)
        push(stack, str[i]);
    for (i = 0; i < n; i++)
        str[i] = pop(stack);
}
int main()
{
    char str[] = "community college";
    reverse(str);
    printf("Reversed string is %s", str);
    return 0;
}
```

OUTPUT:

Reversed string is egelloc ytinummoc