1. Write a c program to reverse a string using stack?

```
/*C program to Reverse String using STACK*/

#include <stdio.h>
#include <string.h>

#define MAX 100        /*maximum no. of characters*/

int top=-1;
int i;

/*declaration of string*/
char stack_string[MAX];

/*function to push character (i)*/
void pushChar(char i);

/*function to pop character (i)*/
char popChar(void);

/*function to check stack is empty or not*/
int isEmpty(void);

/*function to check stack is full or not*/
int isFull(void);

int main()
{
   char str[MAX];

   int i;

   printf("Enter a string: ");
   scanf("%[^\n]s",str);

   for(i=0;i<strlen(str);i++)
      pushChar(str[i]);

   for(i=0;i<strlen(str);i++)
      str[i]=popChar();

   printf("Reversed String is: %s\n",str);
```

```
    return 0;
}

/*function definition of pushChar*/
void pushChar(char i)
{
    /*check for full*/
    if(isFull())
    {
        printf("\nStack is FULL !!!\n");
        return;
    }

    /*increase top and push i in stack*/
    top=top+1;
    stack_string[top]=i;
}

/*function definition of popChar*/
 char popChar()
{
    /*check for empty*/
    if(isEmpty())
    {
        printf("\nStack is EMPTY!!!\n");
        return 0;
    }

    /*pop i and decrease top*/
    i = stack_string[top];
    top=top-1;
    return i;
}

/*function definition of isEmpty*/
int isEmpty()
{
    if(top==-1)
        return 1;
    else
        return 0;
}
```

```
/*function definition of isFull*/
int isFull()
{
   if(top==MAX-1)
      return 1;
   else
      return 0;
}
```

```
Enter a string: karthik
Reversed String is: kihtrak



...Program finished with exit code 0
Press ENTER to exit console.
```

2. Write a program for Infix To Postfix Conversion Using Stack?

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

char pop()
{
   if(a == -1)
      return -1;
   else
      return stack[a--];
}
```

```
int priority(char b)
{
    if(b == '(')
        return 0;
    if(b == '+' || b == '-')
        return 1;
    if(b == '*' || b == '/')
        return 2;
}

Int main()
{
    char exp[20];
    char *e, b;
    printf("Enter the equation :");
    scanf("%s",exp);
    e = exp;
    while(*e != '\0')
    {
        if(isalnum(*e))
            printf("%c",*e);
        else if(*e == '(')
            push(*e);
        else if(*e == ')')
        {
            while((b = pop()) != '(')
                printf("%c", b);
        }
        else
        {
            while(priority(stack[a]) >= priority(*e))
                printf("%c",pop());
            push(*e);
        }
        e++;
    }
    while(a != -1)
    {
        printf("%c",pop());
```

```
  }
}
```

```
Enter the equation :a+b+c
ab+c+

...Program finished with exit code 0
Press ENTER to exit console.
```

3. Write a C Program to Implement Queue Using Two Stacks?

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

/* Functions and variables used */
void push1(int);
void push2(int);
int pop1();
int pop2();
void enqueue();
void dequeue();
void display();
void create();
int stack1[100], stack2[100];
int top1 = -1, top2 = -1;
int count = 0;

/* Main Function */
int main()
{
   int choice;
   printf("\nQUEUE USING STACKS IMPLEMENTATION\n\n");
   printf("\n1.ENQUEUE");
   printf("\n2.DEQUEUE");
   printf("\n3.DISPLAY");
   printf("\n4.EXIT");
   printf("\n");
   create();
```

```
    while (1)
    {
        printf("\npick your choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
                enqueue();
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                exit(0);
            default:
                printf("\nInvalid Choice\n");
        }}}

/* Function to initialize top of two stacks*/
void create()
{
    top1 = top2 = -1;
}

/* Function to push an element to stack */
void push1(int element)
{
    stack1[++top1] = element; // Pushing the element to stack1
}

/* Function to pop element from stack */
int pop1()
{
    return(stack1[top1--]); // Pop element from stack1
}

/* Function to push an element on to stack */
void push2(int element)
{
    stack2[++top2] = element; // Pushing the element to stack2
```

```
}

/* Function to pop an element from stack */
int pop2()
{
   return(stack2[top2--]); // pop element from stack2
}

/* Function to enqueue an element into the queue using stack */
void enqueue()
{
   int my_data, j;
   printf("Enter the my data : ");
   scanf("%d", &my_data);
   push1(my_data); // Push data from stack to the queue
   count++;
}

/* Function to dequeue an element from the queue using stack */
void dequeue()
{
   int j;
   for (j = 0;j <= count;j++)
   {
      push2(pop1()); // Pop elements from stack1 and push them to stack2
   }
   pop2(); // Pop the element from stack2 which is the element to be dequeued
   count--;
   for (j = 0;j <= count;j++)
   {
      push1(pop2()); // Push back all the elements from stack2 to stack1
   }}

//Function to display the elements in the queue/
void display()
{
   int j;
   if(top1 == -1)
   {
      printf("\nEMPTY QUEUE\n");
   }
   else
   {
```

```
    printf("\nQUEUE ELEMENTS : ");
    for (j = 0;j <= top1;j++)
    {
        printf(" %d ", stack1[j]);
    }
    printf("\n");
}}
```

```
QUEUE USING STACKS IMPLEMENTATION


1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT

pick your choice : 1
Enter the my data : 1

pick your choice : 1
Enter the my data : 2

pick your choice : 1
Enter the my data : 3

pick your choice : 3

QUEUE ELEMENTS :  1  2  3

pick your choice : 2

pick your choice : 3

QUEUE ELEMENTS :  2  3

pick your choice : 4


...Program finished with exit code 0
Press ENTER to exit console.
```

4. Write a C Program for insertion and deletion of BST?

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node *leftlink;
   struct node *rightlink;
}*root=NULL;
struct node* insert(struct node* root,int i)
{
   if(root==NULL)
   {
      root=(struct node*)malloc(sizeof(struct node));
      root->data=i;
      root->leftlink=root->rightlink=NULL;
      return root;
  }
   else if(root->data>i)
   {
      root->leftlink=insert(root->leftlink,i);
   }
   else if(root->data<i)
   {
      root->rightlink=insert(root->rightlink,i);
   }
   return root;
}
int minimum(struct node* root)
{
   if(root->leftlink==NULL)
   {
      return root->data;
   }
   else
   {
      return minimum(root->leftlink);
   }

}
struct node* delete(struct node* root,int i)
{
   if(root==NULL)
   {
```

```
    return root;
}
else if(root->data>i)
{
    root->leftlink=delete(root->leftlink,i);

}
else if(root->data<i)
{
    root->rightlink=delete(root->rightlink,i);

}
else
{
    if(root->leftlink==NULL && root->rightlink==NULL)
    {
    remove;
    root;
    return NULL;
    }
    else if(root->leftlink==NULL)
    {
        root=root->rightlink;
    }
    else if(root->rightlink==NULL)
    {
        root=root->leftlink;

    }
    else
    {
        int key=minimum(root->rightlink);
        root->data=key;
        root->rightlink=delete(root->rightlink,key);
    }

}
return root;
}
void inorder(struct node *root)
{
    if(root==NULL)
    {
```

```
        return;

    }
    inorder(root->leftlink);
    printf("%d",root->data);
    inorder(root->rightlink);
}
int main()
{
    int a,b,i;
    printf("Input no of elements:");
    scanf("%d",&a);
    for(b=0;b<a;b++)
    {
        printf("Enter the element:");
        scanf("%d",&i);
        root=insert(root,i);

    }
    inorder(root);
    printf("\a");
    printf("Enter the element that has to remove:");
    scanf("%d",&i);
    root=delete(root,i);
    inorder(root);

}
```

```
Input no of elements:6
Enter the element:1
Enter the element:2
Enter the element:3
Enter the element:4
Enter the element:5
Enter the element:6
123456Enter the element that has to remove:4
12356

...Program finished with exit code 0
Press ENTER to exit console.
```