DSA ASSIGNMENT - 5

1)write a c program to reverse a string using stack?

```
#include <stdio.h>
#include <string.h>
#define MAX 100
int top=-1;
int item;
char stack_string[MAX];
void pushChar(char item);
char popChar();
int vacant();
int overflow();
int main()
{
   char str[MAX];
   int i;
   printf("Enter a string: ");
   scanf("%s",str);
   for(i=0;i<strlen(str);i++)
      pushChar(str[i]);

   for(i=0;i<strlen(str);i++)
      str[i]=popChar();
   printf("The Reversed String is: %s\n",str);
   return 0;
}
void pushChar(char item)
{

   if(overflow())
   {
      printf("The Stack is FULL !\n");
      return;
   }
   top++;
   stack_string[top]=item;
}
char popChar()
{

   if(vacant())
   {
      printf("The Stack is EMPTY !\n");
      return 0;
   }
   item = stack_string[top];
```

```
    top--;
    return item;
}
int vacant()
{
    if(top==-1)
        return 1;
    else
        return 0;
}
int overflow()
{
    if(top==MAX-1)
        return 1;
    else
        return 0;
}
```

2)write a program for Infix To Postfix Conversion Using Stack.

```
#include<stdio.h>
char stack[50];
int top = -1;
void push(char z)
{
    stack[++top] =z;
}

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

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

int main()
```

```
{
   char exp[50];
   char *e, z;
   printf("Enter the expression :: ");
   scanf("%s",exp);
   e = exp;
   while(*e != '\0')
   {
      if(by(*e))
         printf("%c",*e);
      else if(*e == '(')
         push(*e);
      else if(*e == ')')
      {
         while((z= pop()) != '(')
            printf("%c", z);
      }
      else
      {
         while(priority(stack[top]) >= priority(*e))
            printf("%c",pop());
         push(*e);
      }
      e++;
   }
   while(top != -1)
   {
      printf("%c",pop());
   }
}
```

3)write a C Program to Implement Queue Using Two Stacks

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

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;
```

```
int main()
{
    int choice;
    printf("Queue using 2 stack implementation\n");
    printf("1.Enqueue\n");
    printf("2.Dequeue\n");
    printf("3.Display\n");
    printf("4.Exit\n");
    printf("\n");
    create();
    while (1)
    {
        printf("\nEnter 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 Entry!\n");
        }}}


void create()
{
    top1 = top2 = -1;
}


void push1(int element)
{
    stack1[++top1] = element;
 }
```

```
int pop1()
{
   return(stack1[top1--]);
}

void push2(int element)
{
   stack2[++top2] = element;
}


int pop2()
{
   return(stack2[top2--]);
}


void enqueue()
{
   int data, i;
   printf("Enter the elements : ");
   scanf("%d", &data);
   push1(data);
   count++;
}


void dequeue()
{
   int i;
   for (i = 0;i <= count;i++)
   {
      push2(pop1());
   }
   pop2();
   count--;
   for (i = 0;i <= count;i++)
   {
      push1(pop2());
   }}


void display()
{
   int i;
   if(top1 == -1)
```

```
    {
        printf("Queue is Empty!\n");
    }
    else
    {
        printf("The elements in Queue:\n");
        for (i = 0;i <= top1;i++)
        {
            printf(" %d ", stack1[i]);
        }
        printf("\n");
}}
```

4) write a c program for insertion and deletion of BST.

```
#include<stdio.h>
#include<stdlib.>
struct node
{
int value;
struct node *left, *right;
};
struct node *newNode(int item)
{
struct node *temp = (struct node *)malloc(sizeof(struct node));
temp->value = item;
temp->left = temp->right = NULL;
return temp;
}
void inorder(struct node *root)
{
if (root != NULL)
{
inorder(root->left);
printf("%d \n", root->value);
inorder(root->right);
}
}
struct node* insert(struct node* node, int value)
{
if (node == NULL) return newNode(value);
if (value < node->value)
node->left = insert(node->left, value);
else if (value > node->value)
node->right = insert(node->right, value);
return node;
```

```
}
struct node * minValueNode(struct node* node)
{
struct node* current = node;
while (current && current->left != NULL)
current = current->left;
return current;
}
struct node* deleteNode(struct node* root, int value)
{
if (root == NULL) return root;
if (value < root->value)
root->left = deleteNode(root->left, value);
else if (value > root->value)
root->right = deleteNode(root->right, value);
else
{
if (root->left == NULL)
{
struct node *temp = root->right;
free(root);
return temp;
}
else if (root->right == NULL)
{
struct node *temp = root->left;
free(root);
return temp;
}
struct node* temp = minValueNode(root->right);
root->value = temp->value;
root->right = deleteNode(root->right, temp->value);
}
return root;
}
int main()
{
struct node *root = NULL;
root = insert(root, 210);
insert(root,55 );
insert(root, 14);
insert(root, 49);
insert(root, 234);
inorder(root);
printf("Delete 55\n");
root = deleteNode(root, 55);
```

```
Root = deleteNode(root,49);
printf("The modified BS transveral tree: \n");
inorder(root);
return 0;
}
```