ASSIGNMENT-5

Nitesh Kumar

AP19110010542

CSE-‘H’

1. write a c program to reverse a string using stack.

#include <stdio.h>

#include <string.h>

#define max 100

int top=-1;

char str[max];

void push(char x){

if(top == max-1){

printf("str overflow");

} else {

str[++top] =x;

}

}

void pop(){

printf("%c", str[top--]);

}

int main(){

int i;

printf("Enter the string:");

scanf("%s",str);

for(i=0; i<strlen(str); i++)

push(str[i]);

for(i=0; i<strlen(str); i++)

pop();

printf("The reversed string is %s",str);

}

Output1

Enter the string:Computer

The reversed string is retupmoC

2. write a program for Infix to Postfix Conversion Using Stack.

#include<stdio.h>

char stack[20];

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;

}

main()

{

char exp[20];

char \*e, x;

printf("Enter the expression :: ");

scanf("%s", exp);

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 ("%c",pop());

push(\*e);

}

e++;

}

}

Output2

Enter the expression you want to convert: a+b-c\*d=e

ab+cde=\*-

Enter the expression you want to convert: a+b-c/e\*f

ab+ce/f\*-

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

#include <stdio.h>

#include <stdlib.h>

struct sNode {

int data;

struct sNode\* next;

};

void push(struct sNode\*\* top\_ref, int new\_data);

int pop(struct sNode\*\* top\_ref);

struct queue {

struct sNode\* stack1;

struct sNode\* stack2;

};

void enQueue(struct queue\* q, int x)

{

push(&q->stack1, x);

}

int deQueue(struct queue\* q)

{

int x;

if (q->stack1 == NULL && q->stack2 == NULL) {

printf("Q is empty");

getchar();

exit(0);

}

if (q->stack2 == NULL) {

while (q->stack1 != NULL) {

x = pop(&q->stack1);

push(&q->stack2, x);

}

}

x = pop(&q->stack2);

return x;

}

void push(struct sNode\*\* top\_ref, int new\_data)

{

struct sNode\* new\_node = (struct sNode\*)malloc(sizeof(struct sNode));

if (new\_node == NULL) {

printf("Stack overflow \n");

getchar();

exit(0);

}

new\_node->data = new\_data;

new\_node->next = (\*top\_ref);

(\*top\_ref) = new\_node;

}

int pop(struct sNode\*\* top\_ref)

{

int res;

struct sNode\* top;

if (\*top\_ref == NULL) {

printf("Stack underflow \n");

getchar();

exit(0);

}

else {

top = \*top\_ref;

res = top->data;

\*top\_ref = top->next;

free(top);

return res;

}

}

int main()

{

struct queue\* q = (struct queue\*)malloc(sizeof(struct queue));

q->stack1 = NULL;

q->stack2 = NULL;

enQueue(q, 5);

enQueue(q, 25);

enQueue(q, 12);

printf("%d ", deQueue(q));

printf("%d ", deQueue(q));

printf("%d ", deQueue(q));

return 0;

}

Output3

5 25 12

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 e)

{

if(root==NULL)

{

root=(struct node\*)malloc(sizeof(struct node));

root->data=e;

root->leftlink=root->rightlink=NULL;

return root;

}

else if(root->data>e)

{

root->leftlink=insert(root->leftlink,e);

}

else if(root->data<e)

{

root->rightlink=insert(root->rightlink,e);

}

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 e)

{

if(root==NULL)

{

return root;

}

else if(root->data>e)

{

root->leftlink=delete(root->leftlink,e);

}

else if(root->data<e)

{

root->rightlink=delete(root->rightlink,e);

}

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 n,i,e;

printf("Enter the number of elements:");

scanf("%d",&n);

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

{

printf("Enter the element:");

scanf("%d",&e);

root=insert(root,e);

}

inorder(root);

printf("\n");

printf("Enter the element that has to remove:");

scanf("%d",&e);

root=delete(root,e);

inorder(root);

}

Output4

Enter the number of elements:8

Enter the element:1

Enter the element:3

Enter the element:5

Enter the element:7

Enter the element:9

Enter the element:2

Enter the element:4

Enter the element:6

12345679

Enter the element that has to remove:9

1234567