Expt no: 7

**PROGRAM: Implementation of Application of Stack: Infix to Postfix Conversion.**

**(Doubly Linked List)**

**Input:**

#include<stdio.h>

#include<stdlib.h>

struct node {

int data;

struct node \*prev;

struct node \*next;

};

struct node \*start=NULL;

//1

void inst\_beg() {

int x;

printf("Enter value to be inserted\n");

scanf("%d",&x);

struct node \*tmp;

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

if (start==NULL){

tmp->data=x;

tmp->prev=NULL;

tmp->next=NULL;

start=tmp;

printf("%d Inserted",x);

}

else {

tmp->data=x;

tmp->prev=NULL;

tmp->next=start;

start->prev=tmp;

start=tmp;

printf("%d Inserted",x);

}

}

//2

void inst\_end() {

struct node \*p;

int x;

printf("Enter element to be inserted");

scanf("%d",&x);

struct node \*tmp;

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

if (start==NULL) {

tmp->data=x;

tmp->prev=NULL;

tmp->next=NULL;

start=tmp;

printf("%d Inserted",x);

}

else {

p=start;

while(p->next!=NULL){

p=p->next;

}

tmp->data=x;

tmp->next=NULL;

tmp->prev=p;

p->next=tmp;

printf("%d Inserted",x);

}

}

//3

int inst\_pos()

{

struct node \*p, \*tmp;

int i,x,pos;

printf("Enter element you want to insert");

scanf("%d", &x);

printf("Enter Position");

scanf("%d",&pos);

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

tmp->data=x;

if (pos==1)

{

tmp->prev=NULL;

tmp->next=start;

start->prev=tmp;

start=tmp;

printf("%d Inserted at %d",x,pos);

return start;

}

p=start;

for (i=1; i<pos-1; p!=NULL,i++) {

p=p->next;

}

if (p==NULL) {

printf("Less than %d", pos);

}

else{

tmp->next=p->next;

tmp->prev=p;

if (p->next!=NULL){

p->next->prev=tmp;

}

p->next=tmp;

printf("%d Inserted at %d\n",x,pos);

}

return start;

}

//4

void inst\_bfr()

{

struct node \*p,\*tmp;

int x,item;

printf("Enter element you want to insert");

scanf("%d", &x);

printf("Enter element before which you want to insert");

scanf("%d",&item);

if (start==NULL){

printf("Empty");

return;

}

if (start->data==item){

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

tmp->data=x;

tmp->next=start;

tmp->prev=NULL;

start->prev=tmp;

start=tmp;

printf("%d inserted before %d",x,item);

return;

}

p=start;

while(p->next!=NULL){

if (p->next->data==item)

{

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

tmp->data=x;

tmp->next=p->next;

tmp->prev=p;

if(p->next!=NULL)

p->next->prev=tmp;

p->next=tmp;

printf("%d inserted before %d",x,item);

return;

}

p=p->next;

}

printf("Invalid element");

}

//5

void inst\_afrt(){

struct node \*p, \*tmp;

int item,x;

printf("Enter element you want to insert");

scanf("%d", &x);

printf("Enter element after which you want to insert");

scanf("%d",&item);

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

if(start->data==item){

tmp->data=x;

tmp->next=start->next;

if(start->next!=NULL)

start->next->prev = tmp;

tmp->prev=start;

start->next=tmp;

return;

}

p=start;

while (p!=NULL)

{

if (p->data==item)

{

tmp->data=x;

tmp->next=p->next;

if (p->next!=NULL){

p->next->prev=tmp;

}

tmp->prev=p;

p->next=tmp;

printf("%d Inserted after %d", x,item);

return;

}

p=p->next;

}

printf("%d Not present",item);

}

//6

void del\_beg(){

struct node \*p;

if (start==NULL){

printf("List is Empty");

return;

}

else if (start->next==NULL){

p=start;

printf("%d Deleted\n",start->data);

start=NULL;

free(p);

return;

}

else {

printf("%d Deleted",start->data);

p=start;

p->next->prev=NULL;

start=p->next;

free(p);

return;

}

}

//7

void del\_end(){

struct node \*p;

if (start==NULL)

{

printf("List is empty");

return;

}

else if (start->next==NULL){

p=start;

printf("%d Deleted",p->data);

start=NULL;

free(p);

return;

}

else{

p=start;

while (p->next->next!=NULL){

p=p->next;

}

printf("%d Deleted",p->next->data);

free(p->next);

p->next=NULL;

return;

}

}

//8

int del\_inbtwn() {

struct node \*p,\*tmp;

int x;

printf("Enter element you want to delete");

scanf("%d", &x);

if (start==NULL){

printf("Empty");

return start;

}

if (start->data==x){

tmp=start;

start=start->next;

free(tmp);

printf("%d Deleted",x);

return start;

}

p=start;

while(p->next!=NULL){

if (p->next->data==x)

{

tmp=p->next;

p->next=tmp->next;

if (p->next!=NULL){

tmp->next->prev=p;

}

free(tmp);

printf("%d Deleted",x);

return start;

}

p=p->next;

}

printf("%d not found",x);

return start;

}

//9

void del\_pos(){

struct node \*p,\*temp;

int pos,i;

if(start==NULL){

printf("empty");

return;

}

printf("Enter position to be deleted :");

scanf("%d",&pos);

if(pos==1){

temp=start;

if(start->next!=NULL)

start->next->prev=NULL;

start=start->next;

printf("Element %d deleted\n",temp->data);

free(temp);

return;

}

p=start;

for(i=1; i<pos && p!=NULL;i++)

p=p->next;

if(p==NULL) {

printf("position %d not present in the list\n",pos);

return;

}

temp=p;

p->prev->next=p->next;

if(p->next!=NULL)

p->next->prev=p->prev;

printf("Element %d deleted\n",temp->data);

free(temp);

}

//10

void count() {

struct node \*p;

int count=0;

p=start;

while(p!=NULL){

p=p->next;

count++;

}

printf("%d elements present",count);

}

//11

void search()

{

struct node \*p;

int pos=1, x;

printf("Enter search elements");

scanf("%d",&x);

p=start;

while(p!=NULL)

{

if(p->data==x){

printf("element %d found at position %d\n",x,pos);

return;

}

p=p->next;

pos++;

}

printf("element not found\n");

}

//12

void reverse()

{

struct node \*p1,\*p2;

p1=start;

p2=p1->next;

p1->next =NULL;

p1->prev=p2;

while(p2!=NULL)

{

p2->prev=p2->next;

p2->next=p1;

p1=p2;

p2=p2->prev;

}

start=p1;

printf("List reversed\n");

return;

}

//13

void display()

{

struct node \*p;

if (start==NULL)

{

printf("Empty List"); }

else

{

p=start;

while (p!=NULL)

{

printf("%d\t", p->data);

p=p->next; }

}

}

int main()

{

int choice;

printf(" 1:Insert at Beginning of List\n 2:Insert at End of List\n 3:Insert at givenposition\n 4:Insert before node\n 5:Insert after node\n 6:Delete 1st Element\n 7:Delete lastElement\n 8:Delete in between the list\n 9:Delete at any position\n 10:Count elements\n 11:Search Element\n 12: Reverse list\n 13:Display List\n 14:Exit");

while(1)

{

printf("\n Enter choice :");

scanf("%d",&choice);

switch(choice) {

case 1:inst\_beg();

break;

case 2:inst\_end();

break;

case 3:inst\_pos();

break;

case 4:inst\_bfr();

break;

case 5:inst\_afrt();

break;

case 6:del\_beg();

break;

case 7:del\_end();

break;

case 8:del\_inbtwn();

break;

case 9:del\_pos();

break;

case 10:count();

break;

case 11:search();

break;

case 12:reverse();

break;

case 13:display();

break;

case 14:exit(0);

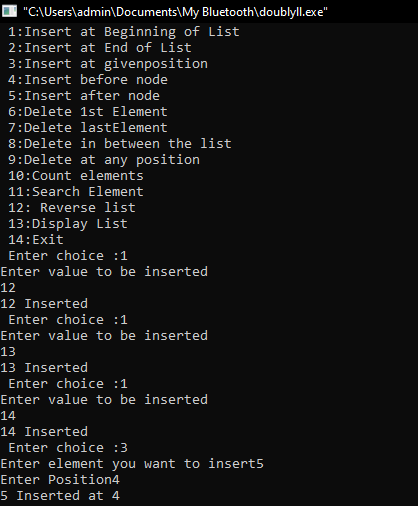
break;

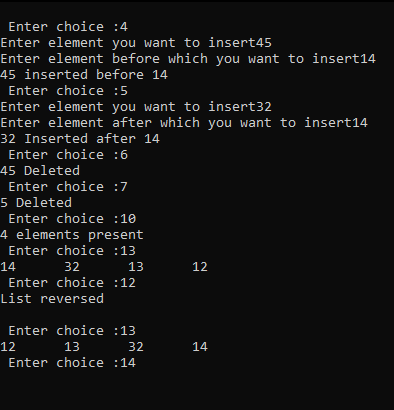
default:printf("Invalid Choice \n"); }

}

return 0;

}

**Output:**



Expt no: 7

**PROGRAM: Implementation of Application of Stack: Infix to Postfix Conversion.**

**(Infix to Postfix)**

**Input:-**

#include<stdio.h>

char stack[100];

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;

return 0;

}

int main()

{

char exp[100];

char \*e, x;

printf("Enter the expression : ");

scanf("%s",exp);

printf("\n");

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++;

}

while(top != -1)

{

printf("%c ",pop());

}return 0;

}

**Output:-**

