1.//Stack implementation using Arrays

#include<stdio.h>

#include<stdlib.h>

#define size 5

int stack[size];

int top = -1;

void push()

{

if(top == size-1)

printf("Stack is Full!");

else

{

int value;

printf("Enter value :");

scanf("%d",&value);

top++;

stack[top] = value;

printf("Value is pushed successfully!\n");

}

}

void pop()

{

if(top == -1)

printf("Stack is Empty!");

else

{

top--;

printf("Value is removed Successfully!\n");

}

}

void peek()

{

if(top == -1)

printf("stack is empty!\n");

else

{

printf("Peek Value : %d\n",stack[top]);

}

}

void display()

{

if(top == -1)

printf("Stack is Empty!\n");

else

{

int i;

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

printf("%d ",stack[i]);

}

}

int main()

{

printf("Stack Implementation:\n");

while(1)

{

int option;

printf("Choose option : 1) push 2) pop 3) peek 4)display 5)exit\nEnter option :");

scanf("%d",&option);

if(option == 1)

push();

else if(option == 2)

pop();

else if(option == 3)

peek();

else if(option == 4)

display();

else

break;

}

return 0;

}

Output: Stack Implementation:

Choose option : 1) push 2) pop 3) peek 4)display 5)exit

Enter option :1

Enter value :23

Value is pushed successfully!

Choose option : 1) push 2) pop 3) peek 4)display 5)exit

Enter option :1

Enter value :43

Value is pushed successfully!

Choose option : 1) push 2) pop 3) peek 4)display 5)exit

Enter option :3

Peek Value : 43

Choose option : 1) push 2) pop 3) peek 4)display 5)exit

Enter option :4

23 43 Choose option : 1) push 2) pop 3) peek 4)display 5)exit

Enter option :2

Value is removed Successfully!

Choose option : 1) push 2) pop 3) peek 4)display 5)exit

Enter option :5

2. #include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX 100

char stack[MAX];

int top = -1;

void push(char x)

{

if (top == MAX - 1)

{

printf("Stack overflow\n");

exit(1);

}

else

{

stack[++top] = x;

}

}

char pop()

{

if (top == -1)

{

printf("Stack underflow\n");

exit(1);

}

else

{

return stack[top--];

}

}

int precedence(char x)

{

if (x == '(')

return 0;

if (x == '+' || x == '-')

return 1;

if (x == '\*' || x == '/')

return 2;

return 0;

}

void infix\_to\_postfix(char infix[], char postfix[])

{

int i = 0;

int j = 0;

char x;

push('(');

strcat(infix, ")");

while (infix[i] != '\0')

{

if (isalnum(infix[i]))

{

postfix[j++] = infix[i++];

}

else if (infix[i] == '(')

{

push(infix[i++]);

}

else if (infix[i] == ')')

{

while ((x = pop()) != '(')

{

postfix[j++] = x;

}

i++;

}

else

{

while (precedence(stack[top]) >= precedence(infix[i]))

{

postfix[j++] = pop();

}

push(infix[i++]);

}

}

postfix[j] = '\0';

}

int main()

{

char infix[MAX];

char postfix[MAX];

printf("Enter the infix expression: ");

scanf("%s", infix);

infix\_to\_postfix(infix, postfix);

printf("The postfix expression is: %s\n", postfix);

return 0;

}

Output: Enter the infix expression: 2+5\*4+(5-2)/3

The postfix expression is: 254\*+52-3/+

3. #include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#define MAX 100

char stack[MAX];

int top = -1;

void push(int x) {

if (top == MAX - 1) {

printf("Stack overflow\n");

exit(1);

} else {

stack[++top] = x;

}

}

int pop() {

if (top == -1) {

printf("Stack underflow\n");

exit(1);

} else {

return stack[top--];

}

}

int evaluate(char x, int op1, int op2) {

switch (x) {

case '+': return op1 + op2;

case '-': return op1 - op2;

case '\*': return op1 \* op2;

case '/':

if (op2 == 0) {

printf("Division by zero error\n");

exit(1);

}

return op1 / op2;

}

return 0;

}

int evaluate\_postfix(char postfix[]) {

int i = 0;

char x;

int op1, op2;

while (postfix[i] != '\0') {

if (isdigit(postfix[i])) {

push(postfix[i] - '0');

} else {

op2 = pop();

op1 = pop();

push(evaluate(postfix[i], op1, op2));

}

i++;

}

return pop();

}

int main() {

char postfix[MAX];

int result;

printf("Enter the postfix expression: ");

scanf("%s", postfix);

result = evaluate\_postfix(postfix);

printf("The result is: %d\n", result);

return 0;

}

Output: Enter the postfix expression: 2+7\*3-(5+3)

Stack underflow

4. #include <stdio.h>

void move(int n, int source, int destination, int intermediate) {

if (n == 1) {

printf("Move disk 1 from shaft %d to shaft %d\n", source, destination);

return;

}

move(n - 1, source, intermediate, destination);

printf("Move disk %d from shaft %d to shaft %d\n", n, source, destination);

move(n - 1, intermediate, destination, source);

}

int main() {

int n = 4;

int source = 1;

int destination = 3;

int intermediate = 2;

printf("Tower of Hanoi solution:\n");

move(n, source, destination, intermediate);

return 0;

}

Output: Tower of Hanoi solution:

Move disk 1 from shaft 1 to shaft 2

Move disk 2 from shaft 1 to shaft 3

Move disk 1 from shaft 2 to shaft 3

Move disk 3 from shaft 1 to shaft 2

Move disk 1 from shaft 3 to shaft 1

Move disk 2 from shaft 3 to shaft 2

Move disk 1 from shaft 1 to shaft 2

Move disk 4 from shaft 1 to shaft 3

Move disk 1 from shaft 2 to shaft 3

Move disk 2 from shaft 2 to shaft 1

Move disk 1 from shaft 3 to shaft 1

Move disk 3 from shaft 2 to shaft 3

Move disk 1 from shaft 1 to shaft 2

Move disk 2 from shaft 1 to shaft 3

Move disk 1 from shaft 2 to shaft 3