DSA EXPERIMENT 4

Harshit Saraswat | 500097101 | B42

**Question 1 :** Using array and functions implement Stack and its operations like insert, delete, and display.

**CODE:**

#include <stdio.h>

#include <stdlib.h>

int \* stack = NULL;

int top = -1;

int main()

{

int repeat = 1;

int size;

printf("Enter Size of Stack : ");

scanf("%d",&size);

stack = (int \*)malloc(size\*sizeof(int));

while (repeat){

printf("Push - 1\nPop - 2\nDisplay - 3\nExit - 4\n");

scanf("%d",&repeat);

switch (repeat){

case 1 : push(size);

break;

case 2 : pop(size);

break;

case 3 : display();

break;

case 4 : return 0;

}

}

return 0;

}

void push(int size){

if (top >= size-1){

printf("\nStack is FULL...Returning\n\n");

return;

}

top++;

printf("Enter New Value To Push : ");

scanf("%d",&stack[top]);

}

void pop(int size){

if (top < 0){

printf("\nStack is Empty...Returning\n\n");

return;

}

printf("\nTop Element : %d\n\n",stack[top]);

top--;

}

void display(){

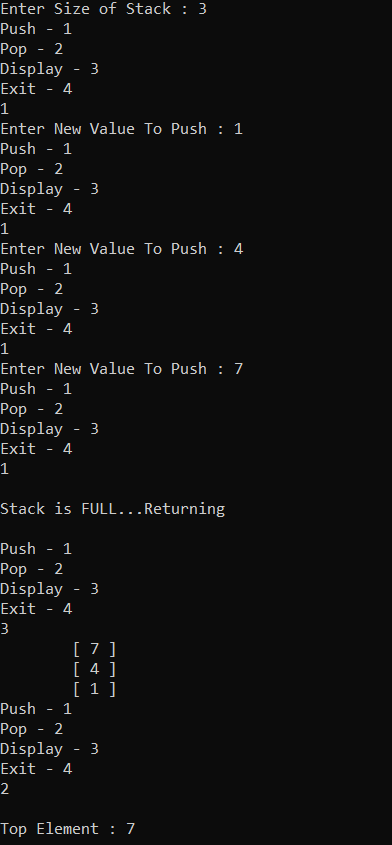
for (int i = top; i >= 0; i--){

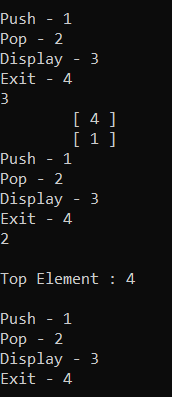
printf("\t[ %d ]\n",stack[i]);

}

}

**OUTPUT:**



****

**QUESTION 2 :** Reverse a string using a stack implemented with Dynamic 1D Array.

**CODE:**

#include <stdio.h>

#include <stdlib.h>

char \* stack = NULL;

int top = -1;

void push (char letter);

void pop();

int main()

{

char stringy [100];

stack = (char \*)malloc(100\*sizeof(char));

printf("Enter a String : ");

scanf("%s",stringy);

char letter = '1';

for (int i = 0 ; letter != '\0'; i++){

letter = stringy[i];

push(letter);

}

top--;

for (top; top != -1; top){

pop();

}

return 0;

}

void push(char letter){

top++;

stack[top] = letter;

}

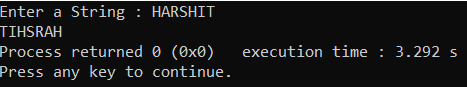
void pop(){

printf("%c",stack[top]);

top--;

}

**OUTPUT:**



**QUESTION 3:** Convert infix to postfix expression using stack implemented using Linked List.

**CODE :**

#include <stdio.h>

#include <stdlib.h>

//INFIX TO POSTFIX

//OPERATORS BY PRECEDENCE \*,/ || +,-

struct node{

char operatoor;

int priority;

struct node \* next;

};

struct node \* top;

struct node \* new\_node;

int return\_priority(char operater);

int main()

{

char infix[50];

printf("Enter Infix Input : ");

scanf("%s",infix);

char element = infix[0];

for(int i = 0; infix[i]!= '\0'; i++,element = infix[i]){

//printf("\nChar PRIORITY = %d\n",return\_priority(element));

if (return\_priority(element) == 0){

printf("%c ",element);

continue;

}

if (element == '('){

push\_to\_top(element);

continue;

}

if (element == ')'){

while(top->operatoor != '('){

printf("%c ",top->operatoor);

pop\_from\_top();

}

pop\_from\_top();

continue;

}

check\_again:

if (top == NULL){

push\_to\_top(element);

continue;

}

if (return\_priority(element) > top->priority){

push\_to\_top(element);

continue;

}

else{

printf("%c ",top->operatoor);

pop\_from\_top();

goto check\_again;

}

}

while(top!=NULL){

printf("%c ",top->operatoor);

pop\_from\_top();

}

return 0;

}

void push\_to\_top(char operater){

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

new\_node->operatoor = operater;

new\_node->priority = return\_priority(operater);

if (top == NULL){

new\_node->next = NULL;

top = new\_node;

return;

}

new\_node->next = top;

top = new\_node;

}

struct node node\_at\_top(){

return \*top;

};

void pop\_from\_top(){

struct node \* temp = top;

top = top->next;

free(temp);

};

int return\_priority(char operater){

if(operater == '+' || operater == '-'){

return 1;

}

else if(operater == '\*' || operater == '/'){

return 2;

}

else if(operater == '(' || operater == ')'){

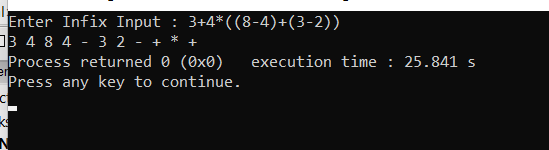
return -1;

}

return 0;

}

**OUTPUT:**

****