**ASSIGNMENT 3**

**Q1**

#include<stdio.h>

#define MAX 60

int stack[MAX];

int top=-1;

int isFull(){

return top== MAX -1;

}

int isEmpty(){

return top == -1;

}

void push(int element){

if (isFull()){

printf("Stack Overflow Condition \n");

return;

}

stack[++top]= element;

printf("%d successfully pushed to stack \n", element);

}

void pop(){

if(isEmpty()){

printf("Stack Underflow Condition \n");

return;

}

printf("%d popped from stack \n", stack[top--]);

}

void peek(){

if(isEmpty()){

printf("Stack is Empty \n");

return;

}

printf("Topmost element of stack is : %d \n",stack[top]);

}

void display(){

if (isEmpty()){

printf("Stack is Empty \n");

return;

}

printf("The elements of Stack are :");

int i;

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

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

}

printf("\n");

}

int main (){

int menu, element;

do {

printf("\n 1)Push \n 2)Pop \n 3)Display \n 4)Peek \n 5)Exit \n");

printf("Enter your menu choice \n");

scanf("%d", &menu);

switch(menu){

case 1:

printf("Enter value to push : ");

scanf("%d", &element);

push(element);

break;

case 2:

pop();

break;

case 3:

display();

break;

case 4:

peek ();

break;

case 5:

printf("Exiting the program \n");

break;

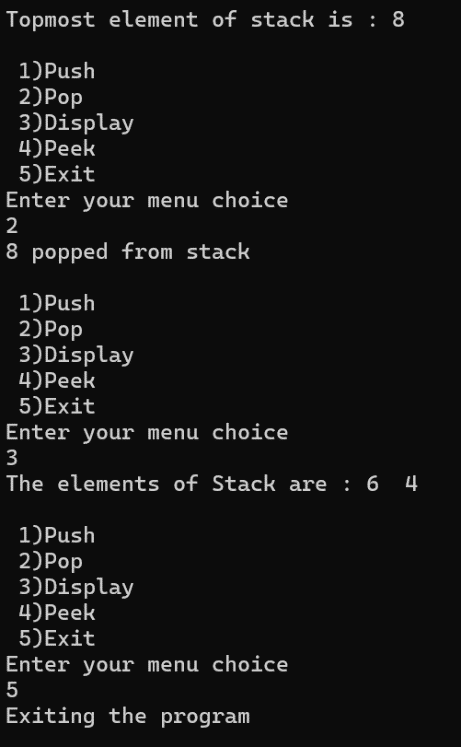
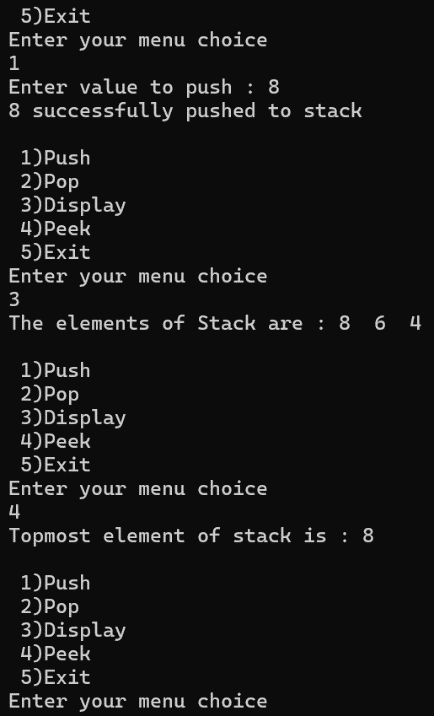
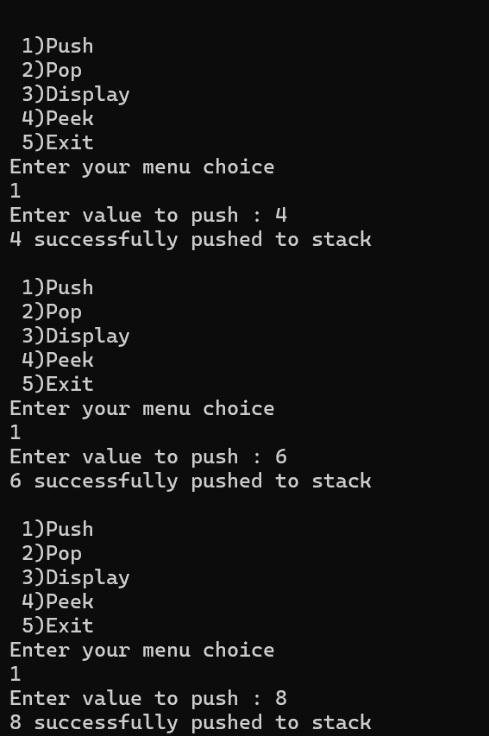
default :

printf("Invalid Input \n");

}

}while (menu!=5);

}

****

**Q2**

#include <stdio.h>

#include <string.h>

#define MAX 40

char stack[MAX];

int top = -1;

void push(char ch){

if (top ==MAX -1){

printf("Stack Overflow Condition \n");

return;

}

stack[++top] =ch;

}

char pop(){

if (top== -1){

printf("Stack Underflow Condition \n");

return '\0';

}

return stack[top--];

}

void reverseString(char str[]){

int n = strlen(str),i;

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

push(str[i]);

}

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

str[i] =pop();

}

}

int main(){

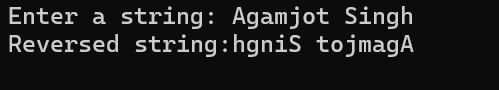
char str[MAX];

printf("Enter a string: ");

gets(str);

reverseString(str);

printf("Reversed string:%s \n",str);

}

**Q3**

#include <stdio.h>

#include <string.h>

#define MAX 50

char stack[MAX];

int top=-1;

void push(char ch){

if (top==MAX-1){

printf("Stack Overflow Condition \n ");

return;

}

stack[++top]=ch;

}

char pop(){

if (top==-1){

printf("Stack Underflow Condition");

return '\0';

}

return stack[top--];

}

int balance(char value[]){

int i;

for(i=0;value[i]!='\0';i++){

if (value[i] == '('||value[i]=='['||value[i]=='{'){

push (value[i]);

}else if ( value [i] ==')'||value[i]=='}'||value[i]=='}'){

char top=pop();

if((value[i]==')'&&top!='(')||(value[i]==']'&&top!='[')||(value[i]=='}'&&top!='{')){

return 0;

}

}

}

return (top==-1);

}

int main (){

char value[MAX];

printf("Enter an expression \n");

fgets(value,60,stdin);

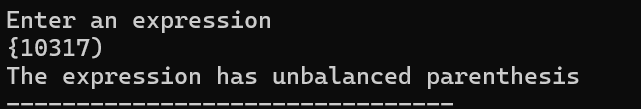
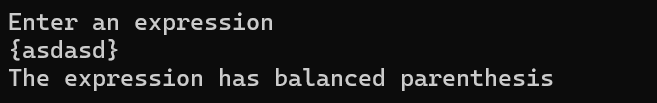
if (balance(value)){

printf("The expression has balanced parenthesis");

}else

printf("The expression has unbalanced parenthesis");

}

****

**Q4**

#include <stdio.h>

#include <ctype.h>

#define MAX 60

char stack[MAX];

int top = -1;

void push(char ch) {

stack[++top] = ch;

}

char pop(){

if (top==-1){

printf("Stack Underflow Condition");

return '\0';

}

return stack[top--];

}

int precedence(char ch) {

switch (ch) {

case '+':

case '-':

return 1;

case '\*':

case '/':

return 2;

case '^':

return 3;

}

return -1;

}

void InToPost(char in[]) {

char post[MAX];

int i = 0, j = 0;

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

if (isalnum(in[i])) {

post[j++] = in[i];

} else if (in[i] == '(') {

push(in[i]);

} else if (in[i] == ')') {

while (top != -1 && stack[top] != '(') {

post[j++] = pop();

}

pop(); // Pop '('

} else {

while (top != -1 && precedence(stack[top]) >= precedence(in[i])) {

post[j++] = pop();

}

push(in[i]);

}

i++;

}

while (top != -1) {

post[j++] = pop();

}

post[j] = '\0';

printf("Postfix Expression: %s\n", post);

}

int main() {

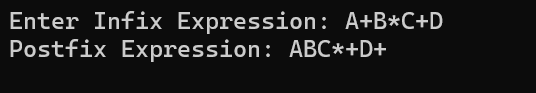
char in[MAX];

printf("Enter Infix Expression: ");

fgets(in,50,stdin);

InToPost(in);

}



**Q5**

#include <stdio.h>

#include <ctype.h>

#include <stdlib.h>

#define MAX 60

int stack[MAX];

int top = -1;

void push(int value) {

if (top == MAX - 1) {

printf("Stack Overflow!\n");

return;

}

stack[++top] = value;

}

int pop() {

if (top == -1) {

printf("Stack Underflow!\n");

exit(1);

}

return stack[top--];

}

int evaluatePostfix(char exp[]) {

int i = 0;

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

if (isdigit(exp[i])) {

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

} else {

int val1 = pop();

int val2 = pop();

switch (exp[i]) {

case '+': push(val2 + val1); break;

case '-': push(val2 - val1); break;

case '\*': push(val2 \* val1); break;

case '/': push(val2 / val1); break;

default:

printf("Invalid operator encountered!\n");

exit(1);

}

}

i++;

}

return pop();

}

int main() {

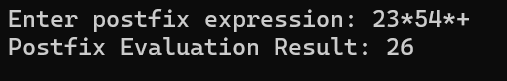
char exp[MAX];

printf("Enter Postfix Expression: ");

gets(exp);

printf("Postfix Evaluation Result: %d\n", evaluatePostfix(exp));

}

****