Assignment 3

Question 1:

#include <stdio.h>

#define SIZE 5

int stack[SIZE], top = -1;

void push(int x) {

if (top == SIZE - 1) printf("Stack is full\n");

else stack[++top] = x;

}

void pop() {

if (top == -1) printf("Stack is empty\n");

else printf("Popped: %d\n", stack[top--]);

}

void isEmpty() {

if (top == -1) printf("Stack is empty\n");

else printf("Stack is not empty\n");

}

void isFull() {

if (top == SIZE - 1) printf("Stack is full\n");

else printf("Stack is not full\n");

}

void display() {

if (top == -1) printf("Stack is empty\n");

else {

printf("Stack: ");

for (int i = top; i >= 0; i--) printf("%d ", stack[i]);

printf("\n");

}

}

void peek() {

if (top == -1) printf("Stack is empty\n");

else printf("Top element: %d\n", stack[top]);

}

int main() {

int ch, x;

while (1) {

printf("\n1.Push 2.Pop 3.isEmpty 4.isFull 5.Display 6.Peek 7.Exit\n");

scanf("%d", &ch);

switch (ch) {

case 1: scanf("%d", &x); push(x); break;

case 2: pop(); break;

case 3: isEmpty(); break;

case 4: isFull(); break;

case 5: display(); break;

case 6: peek(); break;

case 7: return 0;

default: printf("Invalid choice\n");

}

}

}

**Question 2:**

#include <stdio.h>

#include <string.h>

#define SIZE 100

char stack[SIZE];

int top = -1;

void push(char c) {

if (top < SIZE - 1) stack[++top] = c;

}

char pop() {

if (top == -1) return '\0';

return stack[top--];

}

int main() {

char str[SIZE];

printf("Enter a string: ");

scanf("%s", str);

for (int i = 0; i < strlen(str); i++) push(str[i]);

printf("Reversed string: ");

while (top != -1) printf("%c", pop());

return 0;

}

**Question 3:**

#include <stdio.h>

#include <string.h>

#define SIZE 100

char stack[SIZE];

int top = -1;

void push(char c) {

if (top < SIZE - 1) stack[++top] = c;

}

char pop() {

if (top == -1) return '\0';

return stack[top--];

}

int main() {

char exp[SIZE];

printf("Enter expression: ");

scanf("%s", exp);

for (int i = 0; i < strlen(exp); i++) {

if (exp[i] == '(' || exp[i] == '{' || exp[i] == '[') push(exp[i]);

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

char c = pop();

if ((exp[i] == ')' && c != '(') ||

(exp[i] == '}' && c != '{') ||

(exp[i] == ']' && c != '[')) {

printf("Not Balanced\n");

return 0;

}

}

}

if (top == -1) printf("Balanced\n");

else printf("Not Balanced\n");

return 0;

}

**Question 4:**

#include <stdio.h>

#include <ctype.h>

#define SIZE 100

char stack[SIZE];

int top = -1;

void push(char c) {

if (top < SIZE - 1) stack[++top] = c;

}

char pop() {

if (top == -1) return '\0';

return stack[top--];

}

char peek() {

if (top == -1) return '\0';

return stack[top];

}

int precedence(char c) {

if (c == '^') return 3;

if (c == '\*' || c == '/') return 2;

if (c == '+' || c == '-') return 1;

return 0;

}

int main() {

char infix[SIZE], postfix[SIZE];

int i = 0, k = 0;

printf("Enter infix expression: ");

scanf("%s", infix);

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

char c = infix[i];

if (isalnum(c)) postfix[k++] = c;

else if (c == '(') push(c);

else if (c == ')') {

while (top != -1 && peek() != '(') postfix[k++] = pop();

pop();

} else {

while (top != -1 && precedence(peek()) >= precedence(c))

postfix[k++] = pop();

push(c);

}

i++;

}

while (top != -1) postfix[k++] = pop();

postfix[k] = '\0';

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

return 0;

}

**Question 5:**

#include <stdio.h>

#include <ctype.h>

#include <math.h>

#define SIZE 100

int stack[SIZE], top = -1;

void push(int x) {

if (top < SIZE - 1) stack[++top] = x;

}

int pop() {

if (top == -1) return -1;

return stack[top--];

}

int main() {

char exp[SIZE];

printf("Enter postfix expression: ");

scanf("%s", exp);

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

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

else {

int b = pop();

int a = pop();

switch (exp[i]) {

case '+': push(a + b); break;

case '-': push(a - b); break;

case '\*': push(a \* b); break;

case '/': push(a / b); break;

case '^': push(pow(a, b)); break;

}

}

}

printf("Result: %d\n", pop());

return 0;

}