**Assignment 4 Solutions - Stacks and Applications**

**Question 1: Stack using Array**

#include <stdio.h>

#define MAX\_SIZE 5

// Global variables for the stack

int stack[MAX\_SIZE];

int top = -1;

void push() {

int value;

if (top == MAX\_SIZE - 1) {

printf("!! Stack Overflow !!\n");

} else {

printf("Enter the value to push: ");

scanf("%d", &value);

top++;

stack[top] = value;

printf("%d pushed to stack.\n", value);

}

}

void pop() {

if (top == -1) {

printf("!! Stack Underflow !!\n");

} else {

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

top--;

}

}

void display() {

if (top == -1) {

printf("Stack is empty.\n");

} else {

printf("Stack elements are: ");

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

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

}

printf("\n");

}

}

int main() {

int choice;

do {

printf("\n--- Stack Menu (Array) ---\n");

printf("1. Push\n");

printf("2. Pop\n");

printf("3. Display\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1: push(); break;

case 2: pop(); break;

case 3: display(); break;

case 4: printf("Exiting...\n"); break;

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

}

} while (choice != 4);

return 0;

}

Sample Output:

--- Stack Menu (Array) ---

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 1

Enter the value to push: 10

10 pushed to stack.

--- Stack Menu (Array) ---

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 1

Enter the value to push: 20

20 pushed to stack.

--- Stack Menu (Array) ---

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 3

Stack elements are: 20 10

--- Stack Menu (Array) ---

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 2

20 popped from stack.

--- Stack Menu (Array) ---

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 3

Stack elements are: 10

--- Stack Menu (Array) ---

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 4

Exiting...

Question 2: Stack using Linked List

#include <stdio.h>

#include <stdlib.h>

// Node structure

struct Node {

int data;

struct Node\* next;

};

// Global top pointer

struct Node\* top = NULL;

void push() {

int value;

// Create a new node

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

if (newNode == NULL) {

printf("Memory allocation failed. Cannot push.\n");

return;

}

printf("Enter the value to push: ");

scanf("%d", &value);

newNode->data = value;

newNode->next = top; // New node points to the old top

top = newNode; // Top now points to the new node

printf("%d pushed to stack.\n", value);

}

void pop() {

if (top == NULL) {

printf("!! Stack Underflow !!\n");

} else {

struct Node\* temp = top;

printf("%d popped from stack.\n", temp->data);

top = top->next; // Top moves to the next node

free(temp); // Free the popped node's memory

}

}

void display() {

if (top == NULL) {

printf("Stack is empty.\n");

} else {

struct Node\* temp = top;

printf("Stack elements are: ");

while (temp != NULL) {

printf("%d ", temp->data);

temp = temp->next;

}

printf("\n");

}

}

int main() {

int choice;

do {

printf("\n--- Stack Menu (Linked List) ---\n");

printf("1. Push\n");

printf("2. Pop\n");

printf("3. Display\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1: push(); break;

case 2: pop(); break;

case 3: display(); break;

case 4: printf("Exiting...\n"); break;

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

}

} while (choice != 4);

return 0;

}

Sample Output:

--- Stack Menu (Linked List) ---

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 1

Enter the value to push: 100

100 pushed to stack.

--- Stack Menu (Linked List) ---

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 1

Enter the value to push: 200

200 pushed to stack.

--- Stack Menu (Linked List) ---

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 3

Stack elements are: 200 100

--- Stack Menu (Linked List) ---

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 2

200 popped from stack.

--- Stack Menu (Linked List) ---

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 4

Exiting...

Question 3: Infix to Postfix Conversion

#include <stdio.h>

#include <ctype.h>

#include <string.h>

char stack[100];

int top = -1;

void push(char x) {

stack[++top] = x;

}

char pop() {

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

return stack[top--];

}

int precedence(char symbol) {

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

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

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

return 0;

}

int main() {

char infix[100], postfix[100];

int i = 0, j = 0;

char item, x;

printf("Enter the infix expression: ");

scanf("%s", infix);

push('('); // Push '(' onto stack

strcat(infix, ")"); // Add ')' to the end of infix

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

item = infix[i];

if (isalnum(item)) { // If operand, add to postfix

postfix[j++] = item;

} else if (item == '(') {

push(item);

} else if (item == ')') {

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

postfix[j++] = x;

}

} else { // If operator

while (precedence(stack[top]) >= precedence(item)) {

postfix[j++] = pop();

}

push(item);

}

i++;

}

postfix[j] = '\0'; // Null terminate the postfix string

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

return 0;

}

Sample Output:

Enter the infix expression: a+b\*(c^d-e)

Postfix expression: abcd^e-\*+

Question 4: Infix to Prefix Conversion

#include <stdio.h>

#include <string.h>

#include <ctype.h>

char stack[100];

int top = -1;

void push(char x) {

stack[++top] = x;

}

char pop() {

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

return stack[top--];

}

int precedence(char symbol) {

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

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

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

return 0;

}

void reverse(char \*str) {

int len = strlen(str);

for (int i = 0; i < len / 2; i++) {

char temp = str[i];

str[i] = str[len - i - 1];

str[len - i - 1] = temp;

}

}

int main() {

char infix[100], postfix[100], prefix[100];

int i = 0, j = 0;

char item, x;

printf("Enter the infix expression: ");

scanf("%s", infix);

// 1. Reverse the infix string and swap parentheses

reverse(infix);

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

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

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

}

// 2. Find the postfix of the reversed string

i = 0;

push('(');

strcat(infix, ")");

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

item = infix[i];

if (isalnum(item)) {

postfix[j++] = item;

} else if (item == '(') {

push(item);

} else if (item == ')') {

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

postfix[j++] = x;

}

} else {

while (precedence(stack[top]) >= precedence(item)) {

postfix[j++] = pop();

}

push(item);

}

i++;

}

postfix[j] = '\0';

// 3. Reverse the postfix string to get the prefix

strcpy(prefix, postfix);

reverse(prefix);

printf("Prefix expression: %s\n", prefix);

return 0;

}

Sample Output:

Enter the infix expression: a+b\*(c-d)

Prefix expression: +a\*b-cd

Question 5: Postfix Expression Evaluation

#include <stdio.h>

#include <ctype.h>

int stack[100];

int top = -1;

void push(int x) {

stack[++top] = x;

}

int pop() {

return stack[top--];

}

int main() {

char postfix[100];

char \*e;

int n1, n2, n3, num;

printf("Enter the postfix expression: ");

scanf("%s", postfix);

e = postfix;

while (\*e != '\0') {

if (isdigit(\*e)) {

num = \*e - '0'; // Convert char digit to int

push(num);

} else { // It's an operator

n1 = pop();

n2 = pop();

switch (\*e) {

case '+': n3 = n2 + n1; break;

case '-': n3 = n2 - n1; break;

case '\*': n3 = n2 \* n1; break;

case '/': n3 = n2 / n1; break;

}

push(n3);

}

e++;

}

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

return 0;

}

Sample Output:

Enter the postfix expression: 23\*45+

Result of evaluation: 15