Assignment-5

UCS540 (Data Structures and Algorithms)

Submitted by, Harpartap Singh, 102104119

Group- 3EE3

1. **Write a menu driven program with 4 options (Push, Pop, Display, and Exit) to demonstrate the working of stacks using arrays.**

Code:

```
#include <iostream>

using namespace std;

#define MAX 10

int stack[MAX];

int top = -1;

void push(int value) {

if (top >= MAX - 1) {

cout << "Stack overflow. Cannot push " << value << " to the stack.\n";

return;}

stack[++top] = value;

cout << "Pushed " << value << " to the stack.\n";}

int pop() { if (top < 0) {

cout << "Stack underflow.\n";

return -1;}

return stack[top--];}

void display() {

if (top < 0) {

cout << "Stack is empty.\n";

return;}

cout << "Stack elements are: ";

for (int i = top; i >= 0; i--){cout << stack[i] << " ";}

cout << "\n";}
```

```
int main(){int choice;
int value;
do {cout << "Stack Operations Using Arrays"<<endl;
cout << "1. Push"<<endl;
cout << "2. Pop"<<endl;
cout << "3. Display"<<endl;
cout << "4. Exit"<<endl;
cout << "Enter your choice: ";
cin >> choice;
switch (choice) {case 1:
cout << "Enter value to push: ";
cin >> value;
push(value);
break;
case 2: value = pop();
if (value != -1) {cout << "Popped value: " << value << "\n";
} else{cout << "Stack is empty.\n";}
break;
case 3: display();
break;
case 4: cout << "Exiting...\n";
break;
default: cout << "Invalid choice. Please try again.\n";}}
while (choice != 4);}
```

Output:

```
Stack Operations Using Arrays
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter value to push: 3
Pushed 3 to the stack.
Stack Operations Using Arrays
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter value to push: 2
Pushed 2 to the stack.
Stack Operations Using Arrays
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3
Stack elements are: 2 3
```

---

2. **Write a menu driven program with 4 options (Push, Pop, Display, and Exit) to demonstrate the working of stacks using linked-list.**

Code:

```
#include <iostream>

using namespace std;

class Node {public:

int data;

Node* next;};

class Stack {private:

Node* top;

public:

Stack() { top = NULL; }

void push(int val);

void pop();

void display();

bool isEmpty();};

void Stack::push(int val) {

Node* newnode = new Node();

if (!newnode) {cout << "Heap overflow" << endl;

return;}
```

```
newnode->data = val;
newnode->next = top;
top = newnode;}
void Stack::pop() {
if (isEmpty()) {
cout << "Stack underflow" << endl;
return;}
Node* temp = top;
top = top->next;
delete temp;}
void Stack::display() {
if (isEmpty()) {
cout << "Stack is empty" << endl;
return;}
Node* temp = top;
while (temp != NULL) {
cout << temp->data << "->";
temp = temp->next;}
cout << "NULL" << endl;}
bool Stack::isEmpty() {
return top == NULL;}
int main(){Stack stack;
int choice, value;
do{ cout << "Stack Operations Using Arrays"<<endl;
cout << "1. Push"<<endl;
cout << "2. Pop"<<endl;
cout << "3. Display"<<endl;
cout << "4. Exit"<<endl;
cout << "Enter your choice: ";
cin >> choice;
switch (choice) {
```

```
case 1: cout << "Enter value to be pushed: ";

cin >> value;

stack.push(value);

break;

case 2: stack.pop();

break;

case 3: stack.display();

break;

case 4: cout << "Exiting" << endl;

break;

default: cout << "Invalid choice. Please try again." << endl;

break;}

} while (choice != 4);}
```

Output:

```
Enter your choice: 1
Enter value to be pushed: 2
Stack Operations Using Arrays
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3
2->NULL
```

---

3. **Write a program to convert infix expression into postfix expression using stack.**

Code:

```
#include<stdio.h>

#include<ctype.h>

char stack[100];

int top = -1;

void push(char x)

{ stack[++top] = x;}char pop(){if(top == -1)

return -1;
```

```
else
return stack[top--];}
int priority(char x)
{  if(x == '(')
return 0;
if(x == '+' || x == '-')
return 1;
if(x == '*' || x == '/')
return 2;
return 0;
}int main(){ char exp[100];
char *e, x;
printf("Enter the expression : ");
scanf("%s",exp);
printf("\n");
e = exp;
while(*e != '\0')
{  if(isalnum(*e))
printf("%c ",*e);
else if(*e == '(')
push(*e);
else if(*e == ')'){while((x = pop()) != '(')
printf("%c ", x);}
else{while(priority(stack[top]) >= priority(*e))
printf("%c ",pop());
push(*e);}
e++;}
while(top != -1)
{printf("%c ",pop());
}return 0;}
```

Output:

```
Enter the expression : a+b*c

a b c * +
```

---

4. **Write a program to convert infix expression into prefix expression using stack.**

Code:

```
#include <iostream>

#include <stack>

#include <string>

#include <algorithm>

using namespace std;

bool isOperator(char c) {

return (!isalpha(c) && !isdigit(c));}

int getPriority(char C) {

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

return 1;

else if (C == '*' || C == '/')

return 2;

else if (C == '^')

return 3;

return 0;}

string infixToPostfix(string infix) {

infix = '(' + infix + ')';

int l = infix.size();

stack<char> char_stack;

string output;

for (int i = 0; i < l; i++) {

if (isalpha(infix[i]) || isdigit(infix[i]))

output += infix[i];

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

```
char_stack.push('(');
else if (infix[i] == ')') {
while (char_stack.top() != '(') {
output += char_stack.top();
char_stack.pop();
}
char_stack.pop();
}
else {
if (isOperator(char_stack.top())) {
if (infix[i] == '^') {
while (getPriority(infix[i]) <= getPriority(char_stack.top())) {
output += char_stack.top();
char_stack.pop();
}
}
else {
while (getPriority(infix[i]) < getPriority(char_stack.top())) {
output += char_stack.top();
char_stack.pop();
}
}
char_stack.push(infix[i]);
}
}
}
while (!char_stack.empty()) {
output += char_stack.top();
char_stack.pop();
}
return output;
```

```
}
string infixToPrefix(string infix) {
int l = infix.size();
reverse(infix.begin(), infix.end());
for (int i = 0; i < l; i++) {
if (infix[i] == '(') {
infix[i] = ')';
}
else if (infix[i] == ')') {
infix[i] = '(';
}
}
string prefix = infixToPostfix(infix);
reverse(prefix.begin(), prefix.end());
return prefix; } int main(){string s;
cout << "Enter an infix expression: ";
getline(cin, s);
cout << "Prefix expression: " << infixToPrefix(s) << endl;}
```

Output:

```
Enter an infix expression: a+(b*c)/d
Prefix expression: +a/*bcd
```

---

5. **Write a program to evaluate the postfix expression using stack.**

Code:

```
#include<stdio.h>
int stack[20];
int top = -1;
void push(int x){stack[++top] = x;}
int pop(){return stack[top--];}
int main(){char exp[20];
char *e;
```

```
int n1,n2,n3,num;
printf("Enter the expression :: ");
scanf("%s",exp);
e = exp;
while(*e != '\0'){if(isdigit(*e)){num = *e - 48;
push(num);}
else{n1 = pop();
n2 = pop();
switch(*e)
{case '+':
{n3 = n1 + n2;
break;}
case '-':
{n3 = n2 - n1;
break;}
case '*':
{n3 = n1 * n2;
break;}
case '/':
{n3 = n2 / n1;
break;}}
push(n3);}e++;}
printf("The result of expression %s  =  %d",exp,pop());
return 0;}
```

Output:

```
Enter the expression :: 245+*
The result of expression 245+*  =  18
```
