LAB REPORT

CSE 114 : Data Structure and Algorithms Sessional

PREPARED BY

Mehrin Farzana
ID: 2101013
Session: 2021-2022
Date: 22/09/2023

SUPERVISED BY

Suman Saha
Lecturer
Department of IRE, BDU

BANGABANDHU SHEIKH MUJIBUR RAHMAN DIGITAL UNIVERSITY
(BDU)

List of Problems

1. Implement stack using array.

2. Evaluation of Postfix Expressions Using Stack.

3. C Program to Convert Infix to Postfix Expression using Stack.

Problem No.: 01

Problem Statement:

Implement stack using array.

Code:

```
#include<stdio.h>
#include<stdbool.h>


int top = -1;
int push(int *a, int n, int el){
    if(top==n-1){
        printf("Overflow!\n");
        return -1;
    }
    a[++top]=el;
    return 0;
}

int pop(int *a){
    if(top==-1){
        return -1;
    }
    top--;
    return a[top+1];
}

int peek(int *a){
    if(top==-1){
        printf("Empty stack\n");
        return -1;
    }
    return a[top];
}

bool isFull(int n){
    if(top==n-1){
        return true;
    }
    return false;
}
```

```
bool isEmpty(){
   if(top==-1){
      return true;
   }
   return false;
}

void display(int a[]){
   printf("Your stack:\n");
   for(int i=top; i>-1; i--){
      printf("%d\n", a[i]);
   }
}

int main(){

   int n, top = -1, chc=1, el;
   char c;
   printf("Size of stack: ");
   scanf("%d", &n);
   int a[n];
   while(chc){
      printf("Menu:\n1. PUSH\n2. POP\n3. TopElement\n4. Display\nEnter your
choice: ");
      scanf("%d", &chc);
      if(chc==1){
         printf("Element to insert: ");
         scanf("%d", &el);
         push(a, n, el);

      }
      else if(chc==2){
         int tmp = pop(a);
         if(tmp!=-1){
            printf("Popped element: %d\n", tmp);
         }
         else
            printf("Underflow!\n");;

      }
      else if(chc==3){
         printf("Top element: %d\n", peek(a));
      }
      else if(chc==4){
         display(a);
      }
      printf("Do you wish to continue?(y/n) ");
      scanf(" %c", &c);
      if(c=='n')
         chc=0;
```

```
    }

    return 0;
}
```

Output:

```
Size of stack: 5
Menu:
1. PUSH
2. POP
3. TopElement
4. Display
Enter your choice: 1
Element to insert: 3
Do you wish to continue?(y/n) y
Menu:
1. PUSH
2. POP
3. TopElement
4. Display
Enter your choice: 4
Your stack:
3
Do you wish to continue?(y/n) y
Menu:
1. PUSH
2. POP
3. TopElement
4. Display
Enter your choice: 1
Element to insert: 34
Do you wish to continue?(y/n) y
```

```
Do you wish to continue?(y/n) y
Menu:
1. PUSH
2. POP
3. TopElement
4. Display
Enter your choice: 4
Your stack:
34
3
Do you wish to continue?(y/n) y
Menu:
1. PUSH
2. POP
3. TopElement
4. Display
Enter your choice: 3
Top element: 34
Do you wish to continue?(y/n) y
Menu:
1. PUSH
2. POP
3. TopElement
4. Display
Enter your choice: 1
Element to insert: 4
Do you wish to continue?(y/n) y
```

```
Do you wish to continue?(y/n) y
Menu:
1. PUSH
2. POP
3. TopElement
4. Display
Enter your choice: 1
Element to insert: 6
Do you wish to continue?(y/n) y
Menu:
1. PUSH
2. POP
3. TopElement
4. Display
Enter your choice: 1
Element to insert: 8
Do you wish to continue?(y/n) y
Menu:
1. PUSH
2. POP
3. TopElement
4. Display
Enter your choice: 1
Element to insert: 9
Overflow!
Do you wish to continue?(y/n) y
```

```
Menu:
1. PUSH
2. POP
3. TopElement
4. Display
Enter your choice: 4
Your stack:
8
6
4
34
3
Do you wish to continue?(y/n) y
Menu:
1. PUSH
2. POP
3. TopElement
4. Display
Enter your choice: 2
Popped element: 8
Do you wish to continue?(y/n) y
Menu:
1. PUSH
2. POP
3. TopElement
4. Display
```

```
Enter your choice: 4
Your stack:
6
4
34
3
Do you wish to continue?(y/n) y
Menu:
1. PUSH
2. POP
3. TopElement
4. Display
Enter your choice: 2
Popped element: 6
Do you wish to continue?(y/n) y
Menu:
1. PUSH
2. POP
3. TopElement
4. Display
Enter your choice: 2
Popped element: 4
Do you wish to continue?(y/n) y
```

```
Menu:
1. PUSH
2. POP
3. TopElement
4. Display
Enter your choice: 2
Popped element: 34
Do you wish to continue?(y/n) y
Menu:
1. PUSH
2. POP
3. TopElement
4. Display
Enter your choice: 2
Popped element: 3
Do you wish to continue?(y/n) y
Menu:
1. PUSH
2. POP
3. TopElement
4. Display
Enter your choice: 2
Underflow!
Do you wish to continue?(y/n) y
Menu:
1. PUSH
2. POP
3. TopElement
4. Display
Enter your choice: 2
Underflow!
Do you wish to continue?(y/n) n
PS C:\Users\BAB AL SAFA\OneDrive\Desktop\inC>
```

Fig 1.1: Output on console for case 1.

Problem No.: 02

Problem Statement:

Evaluation of Postfix Expressions Using Stack.

Code:

```
#include<stdio.h>
#include<stdbool.h>
#include<string.h>


int top = -1;
int push(int *a, int n, int el){
   if(top==n-1){
      printf("Overflow!\n");
      return -1;
   }
   a[++top]=el;
   return 0;
}

int pop(int *a){
   if(top==-1){
      return -1;
   }
   top--;
   return a[top+1];
}

int peek(int *a){
   if(top==-1){
      printf("Empty stack\n");
      return -1;
   }
   return a[top];
}

bool isFull(int n){
   if(top==n-1){
      return true;
   }
   return false;
}
```

```
bool isEmpty(){
    if(top==-1){
        return true;
    }
    return false;
}

void display(int a[]){
    printf("Your stack:\n");
    for(int i=top; i>-1; i--){
        printf("%d\n", a[i]);
    }
}


int char_to_int(char a[]){
    int b[10],i,j,sum=0, mul=1;
    for(i=strlen(a)-1; i>=0; i--){
        b[i]=(a[i]-'0')*mul;
        mul*=10;
    }
    for(j=0; j<strlen(a); j++){
        sum+=b[j];
    }

    return sum;
}

int main(){
    char c[101];
    int a[101], op1, op2;
    int num=0;

    for(int i=0; ; i++){
        scanf("%c", &c[i]);
        if(c[i]=='\n') break;
        if(c[i]!=' ' && c[i]!='+' && c[i]!='-' && c[i]!='*' && c[i]!='/' && c[i]!='^'){
            num=(num*10)+(int)(c[i]-'0');
            printf("%d\n", num);

        }
        else if(c[i]==' ' && c[i-1]!='+' && c[i-1]!='-' && c[i-1]!='*' && c[i-1]!='/' &&
c[i-1]!='^'){
            push(a, 101, num);
            num=0;
        }
        else if(c[i]=='+' || c[i]=='-' || c[i]=='*' || c[i]=='/' || c[i]=='^'){
            op2=pop(a);
            op1=pop(a);
```

```
        if(c[i]=='+')
            push(a, 101, op1+op2);
        else if(c[i]=='-')
            push(a, 101, op1-op2);
        else if(c[i]=='*')
            push(a, 101, op1*op2);
        else if(c[i]=='/')
            push(a, 101, op1/op2);
        else if(c[i]=='^')
            push(a, 101, op1^op2);
    }
  }

  printf("%d\n", pop(a));
  return 0;
}
```

Output:

```
33 22 +
55
```

Fig 1.1: Output on console for case 1.

```
100 200 +2 / 5 * 7 +
757
```

Fig 1.2: Output on console for case 2.

```
5 1 2 + 4 * + 3 -
14
```

Fig 1.3: Output on console for case 3.

```
1 2 + 3 * 4 5 6 + / -
9
```

Fig 1.4: Output on console for case 4.

Problem No.: 03

Problem Statement:

C Program to Convert Infix to Postfix Expression using Stack.

Code:

```
// C code to convert infix to postfix expression

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_EXPR_SIZE 100

int precedence(char operator)
{
	switch (operator) {
	case '+':
	case '-':
		return 1;
	case '*':
	case '/':
		return 2;
	case '^':
		return 3;
	default:
		return -1;
	}
}

int isOperator(char ch)
{
	return (ch == '+' || ch == '-' || ch == '*' || ch == '/'
				|| ch == '^');
}

char* infixToPostfix(char* infix)
{
	int i, j;
	int len = strlen(infix);
	char* postfix = (char*)malloc(sizeof(char) * (len + 2));
	char stack[MAX_EXPR_SIZE];
	int top = -1;
```

```
	for (i = 0, j = 0; i < len; i++) {
		if (infix[i] == ' ' || infix[i] == '\t')
			continue;
		if (isalnum(infix[i])) {
			postfix[j++] = infix[i];
		}

		else if (infix[i] == '(') {
			stack[++top] = infix[i];
		}
		else if (infix[i] == ')') {
			while (top > -1 && stack[top] != '(')
				postfix[j++] = stack[top--];
			if (top > -1 && stack[top] != '(')
				return "Invalid Expression";
			else
				top--;
		}

		else if (isOperator(infix[i])) {
			while (top > -1
				&& precedence(stack[top])
					>= precedence(infix[i]))
				postfix[j++] = stack[top--];
			stack[++top] = infix[i];
		}
	}

	while (top > -1) {
		if (stack[top] == '(') {
			return "Invalid Expression";
		}
		postfix[j++] = stack[top--];
	}
	postfix[j] = '\0';
	return postfix;
}

int main()
{
	char infix[MAX_EXPR_SIZE];
   scanf("%s", infix);
	char* postfix = infixToPostfix(infix);
	printf("%s\n", postfix);
	free(postfix);
	return 0;
}
```

Output:

```
a+b-c*d/h^f
ab+cd*hf^/-
```

Fig 1.1: Output on console for case 1.

```
a+b+c-b*v^h^f^g
ab+c+bvh^f^g^*-
```

Fig 1.2: Output on console for case 2.