Practical-4

Stack Operations & Application

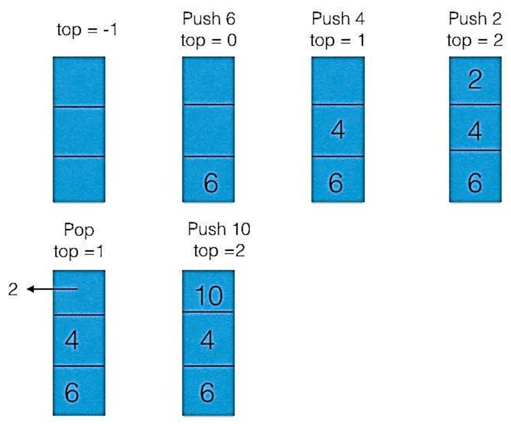
Stack is a linear data structure that follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out).

Mainly the following three basic operations are performed in the stack:

- **Push:** Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.
- **Pop:** Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.
- **Peek or Top:** Returns the top element of the stack.
- **isEmpty:** Returns true if the stack is empty, else false.

A stack can be implemented in C language using:

- Array
- Linked List



Implementation of Stack using Array

Implementation of Stack using Array

```
#include<stdio.h>
#include<conio.h>
#define SIZE 10
void push(int);
void pop();
void display();
int stack[SIZE], top = -1;
void main()
{   int value, choice;
   clrscr();
   while(1){
      printf("\n\n***** MENU *****\n");
      printf("1. Push\n2. Pop\n3. Display\n4. Exit");
      printf("\nEnter your choice: ");
      scanf("%d",&choice);
      switch(choice){
       case 1: printf("Enter the value to be insert: ");
            scanf("%d",&value);
            push(value);
            break;
       case 2: pop();
            break;
       case 3: display();
            break;
       case 4: exit(0);
       default: printf("\nWrong selection!!! Try again!!!");
      }
   }
}
void push(int value){
   if(top == SIZE-1)
      printf("\nStack is Full!!! Insertion is not possible!!!");
   else{
      top++;
      stack[top] = value;
      printf("\nInsertion success!!!");
   }
}
void pop(){
   if(top == -1)
      printf("\nStack is Empty!!! Deletion is not possible!!!");
   else{
      printf("\nDeleted : %d", stack[top]);
      top--;
   }
}
void display(){
   if(top == -1)
      printf("\nStack is Empty!!!");
   else{
      int i;
      printf("\nStack elements are:\n");
      for(i=top; i>=0; i--)
       printf("%d\n",stack[i]);
   }
}
```

Evaluation of Postfix Expressions

The compiler finds it convenient to evaluate an expression in its postfix form. The virtues of postfix form include elimination of parentheses which signify priority of evaluation and the elimination of the need to observe rules of hierarchy, precedence and associativity during evaluation of the expression.

As **Postfix expression** is without parenthesis and can be evaluated as two operands and an operator at a time, this becomes easier for the compiler and the computer to handle.

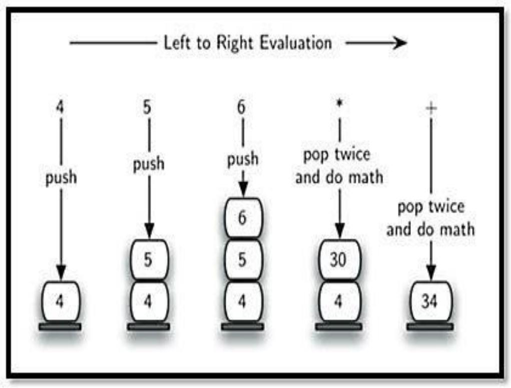
Evaluation rule of a Postfix Expression states:

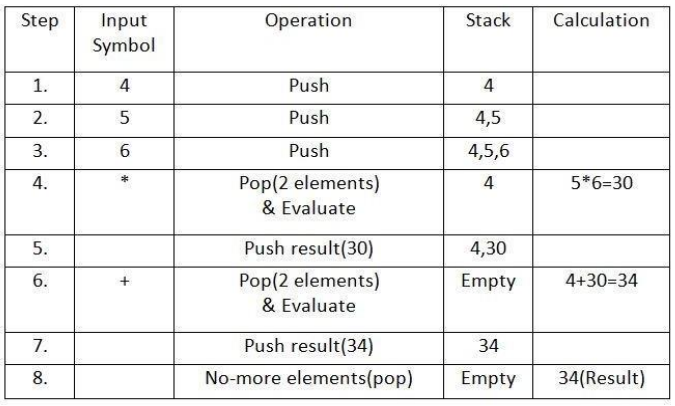
1. While reading the expression from left to right, push the element in the stack if it is an operand.
2. Pop the two operands from the stack, if the element is an operator and then evaluate it.
3. Push back the result of the evaluation. Repeat it till the end of the expression.

Algorithm

```
1) Add ) to postfix expression.
2) Read postfix expression Left to Right until ) encountered
3) If operand is encountered,
   push it onto Stack [End If]
4) If operator is encountered, Pop two elements
   i) A -> Top element
   ii) B-> Next to Top element
   iii) Evaluate B operator A
   iv) push B onto Stack
5) Set result = pop
6) END
```

Example Expression: 456*+





Step	Input Symbol	Operation	Stack	Calculation
1.	4	Push	4	
2.	5	Push	4,5	
3.	6	Push	4,5,6	
4.	*	Pop(2 elements) & Evaluate	4	5*6=30
5.		Push result(30)	4,30	
6.	+	Pop(2 elements) & Evaluate	Empty	4+30=34
7.		Push result(34)	34	
8.		No-more elements(pop)	Empty	34(Result)

Result: 34

The following algorithm converts infix to postfix.

- Scan input string from left to right character by character.
- If the character is an operand, put it into output stack.
- If the character is an operator and operator's stack is empty, push operator into operators' stack.

- If the operator's stack is not empty, there may be following possibilities.
    - If the precedence of scanned operator is greater than the top most operator of operator's stack, push this operator into operand's stack.
    - If the precedence of scanned operator is less than or equal to the top most operator of operator's stack, pop the operators from operand's stack until we find a low precedence operator than the scanned character. Never pop out (**'('**) or (**')'**) whatever may be the precedence level of scanned character.
    - If the character is opening round bracket (**'('**), push it into operator's stack.
    - If the character is closing round bracket (**')'**), pop out operators from operator's stack untill we find an opening bracket (**'('**).
    - Now pop out all the remaining operators from the operator's stack and push into output stack.

```
#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;
}
```

Exercise

1. Write a program to implement following stack operations using array with MAX elements. Use Character array
   - PUSH
   - POP
   - PEEP
   - CHANGE
   - DISPLAY
   - IS_FULL
   - IS_EMPTY
2. Write a program to evaluate a postfix expression.
3. Write a program for infix to postfix conversion of an expression.
4. Write a program to reverse the string using stack operations