Lab - 3

Page No.
Date 01 / 01 / 2024

1. Write a program to simulate the working of stack using an array with the following:

a) PUSH
b) POP
c) DISPLAY

The program should print appropiate messages for stack overflow, stack underflow.

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 5
void push();
void pop();
void display();
int stack[SIZE], top = -1;

int main()
{
 int choice;
 while()
 {
   printf("\n perform stack operations:");
   printf("\n 1. push \n 2. Pop \n 3. Display \n 4. Exit");
```

```
printf("\n Enter the choice: ");
scanf("%d", &choice);

switch (choice)
{
 case 1: push();
         break;
 case 2: pop();
         break;
 case 3: display();
         break;
 case 4: exit(0);
 default: printf("Invalid choice!!");
}
}
}.
```

```
void push()
{
 int x;
 if (top == SIZE - 1)
{
  printf ("\n overflow ");
}
else
 {
  printf (" Enter the element to be added onto the stack- ");
  scanf ("%d", &x);
  top = top+1;
  stack[top] = x;
}
}.

void pop()
{
 if (top == -1)
{
  printf ("\n Underflow ");
}
else
{
  printf ("\n Element present in the stack: \n");
```

```
printf("\n Popped element : %d", stack[top]);
top = top - 1;
}
}

void ~~show~~ diplplay()
{
 if (top == -1)
 {
 printf ("\n underflow ");
 }
 else
 {
 printf ("\n Elements present in the stack: \n");
 for (int i = top; i >= 0; --i);
  printf ("%d \n", stack[~~top~~ i]);
 }
}
```

output:

perform operation on stack :

1. Push
2. Pop
3. diplplay
4. Exit

Enter the choice = 1

Enter the no = 5

inserted 5.

2) Infix to postfix.

```
void push (char symbol)
{
  top = top + 1;
  stack [top] = symbol;
}

char pop ()
{
 char symb;
 symb = stack [top];
 top = top - 1;
 return (symb);
}

int pred (char symbol)
{
 int p;
switch (symbol)
{
 case '^' : p = 3;
  break;
 case '*' :
 case '/' : p = 2;
```

```
        break;
    case '+':
    case '-': p=1;
        break;
    case '(': p=0;
        break;
    case '#': p=-1;
        break;
    }
    return(p);
}
```

3.

```
void infix_to_postfix()
{
    length = strlen(infix);
    push('#');
    while (index1 < length)
    {
        symbol = infix[index1];
        switch(symbol)
        {
        case '(': push(symbol);
            break;
        case ')': temp = pop();
            while (temp != '(')
            {
```

```
        postfix[pos] = itemp;
        pos++;
        temp = pop();
        }
        break;
case '+':
case '-':
case '*':
case '/':
case '^': while (pred(stack[top]) >= pred(symbol))
            {
            temp = pop();
            postfix[pos++] = temp;
            }
            push(symbol);
            break;
    default : postfix[pos++] = symbol;
  }
  index++;
}
while (top > 0)
{
 temp = pop();
 postfix[pos++] = temp;
}
}.
```

Sun 3/11/24

Output:

Enter Infix Expression:

A+B*C+D

Postfix expression:

ABC*+D+