1/01/2024

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 appropriate message for stack overflow and, stack underflow.

```
# define size 5;
int top=-1, Stack[size];

void push(int e)
 {
      if (top==size)
       {
           printf("Stack overflow");
       }
      else
       {
           top = top+1 ;
           stack[top]=e;
           printf(" The insertion operation is complete");
       }
 }

void pop()
{
    if (top==-1)
     {
        printf("Stack underflow, Stack is empty!!");
     }
    else
     {
        ~~top=top-1~~
        printf(" the deleted element is:%d", stack[top]);
        top= top-1;
 }

void display ()
{
    if (top==-1)
     {
        printf("stack is empty"); }
```

```
else
{
    for (i=top; i>=0; i--)
    {
        printf("%d", stack[i]);
    }
}
}
```

~~int main()~~

~~{~~

~~int value, choice~~

Infix to postfix

```
int index=0, pos=0, top=-1, length;
char symbol, temp, infix[20], postfix[20], stack[20];
~~void~~

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

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

int presidance (char symbol)
{
    int p;
    switch (symbol)
    {
        case '*':
        case '/': p= 2;
                  break;
        case '+': p= 1;
        case '-':
```

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

void infixtopostfix();
{
  length = strlen (infix);
  push('#');
  while (length > index)
  {
      Symbol = infix[index];
      Switch (symbol)
      {
        case '(' : push(symbol);
                   break;
        case ')' : temp = pop()
                   while(temp != '(')
                   {
                     postfix [pos] = temp;
                     pos++;
                     temp = pop();
                   }
                   break;
        Case '+':
        Case '-':
        Case '*':
        Case '/':
                   while (precidance(stack[top]) >= precidance(symbol))
                   {
                     temp = pop();
                     postfix [pos++] = temp;
                   }
                   push(symbol);
                   break;
        default : postfix [pos++] = symbol;
      }
      index++;
```

Sw 1/11/24

```
while (top>0)
{
    temp = pop();
    postfix[pol++] = temp;
```

output:

1. stack:

~~output~~ - - - Menu - - -

1. push.
2. pop
3. display
4. exit

1. → entered 1 - enter a value
   so
   Insertion operation is complete

2. → entered 2 - deleted value 50

3. → entered 3 — stack is empty

4. ~~exit~~ entered 4 — exits the stack.

2. Infix to postfix

① enter a infix expression:
a+b*(c^d-e)^(f+g*h)-i

~~output~~
postfix: abcd^e-fgh*+^*+i-

①

infix: ((A+B)-(C*(D/E))+F

postfix: AB+CDE/*-F+

```c
#include<stdio.h>
#include<stdlib.h>

int size=5;
int top=-1,stack[5];

void push(int a)
{
    if(top==size)
    {
        printf("stack overflow\n");
    }
    else{
        top=top+1;
        stack[top]=a;
        printf("insertion operation is complete\n");
    }
}

void pop()
{
    if (top==-1)
    {
        printf("stack is empty\n");
    }
    else{
        top--;
    }
}

void display()
{
    if (top==-1)
    {
        printf("stack is empty\n");
    }
    else{
```

```
        else{
            for(int i=top;i>=0;i--)
            {
                printf("%d\n",stack[i]);
            }
        }
    }

    int main()
    {
        int value,choice,t=0;
        while(1)
        {
            printf("---------------MENU----------------\n");
            printf("1.push\n 2.pop\n 3.display\n 4.exit\n");
            scanf("%d",&choice);
            switch (choice)
            {
            case 1: printf("enter a value:\n");
                    scanf("%d",&value);
                    push(value);
                break;

            case 2: pop();
                break;

            case 3:display();
                break;

            case 4:exit(0);
                break;

            default:printf("wrong input!\n");
                break;
            }
        }
    }
```

```
----------------MENU----------------
1.push
 2.pop
 3.display
  4.exit
1
enter a value:
50
insertion operation is complete
----------------MENU----------------
1.push
 2.pop
 3.display
 4.exit
3
50
----------------MENU----------------
1.push
 2.pop
 3.display
 4.exit
2
----------------MENU----------------
1.push
 2.pop
 3.display
 4.exit
3
stack is empty
----------------MENU----------------
1.push
 2.pop
 3.display
 4.exit
4

Process returned 0 (0x0)   execution time : 20.880 s
Press any key to continue.
```

3rd_sem > C learning > C infix.c > infixtopostfix()

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

int top=-1,pos=0;
char temp,stack[25],infix[25],postfix[25];

void push(char s)
{
    stack[++top]=s;
}

int precedence(char s)
{
    switch(s)
    {
        case '^':
            return 3;
        case '+':
        case '-':
            return 1;
        case '*':
        case '/':
            return 2;
        case '(':
            return 0;
    }
}

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

void infixtopostfix()
```

```c
void infixtopostfix()
{
    int len=strlen(infix);
    int i=0;
    char symbol;
    while(i<len)
    {
        symbol=infix[i];

        switch(symbol)
        {
            case '(':
                push(symbol);
                break;

            case ')':
                temp=pop();
                while(temp!='(')
                {
                    postfix[pos++]=temp;
                    temp=pop();
                }
                break;

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

```
    }
    while(top!=-1)
    {
        postfix[pos++]=stack[top];
        top--;
    }
    return;
}

int main()
{
    printf("enter a infix problem:\n");
    scanf("%s",infix);
    infixtopostfix();
    printf("infix :%s\n",infix);
    printf("postfix :%s\n",postfix);

}
```

```
enter a infix problem:
a+b(c^d-e)^(f+g*h)-i
infix :a+b(c^d-e)^(f+g*h)-i
postfix :abcd^e-fgh*+^+i-

Process returned 0 (0x0)   exec
```