01/08/2024

WEEK-3

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 appropriate messages for stack overflow, stack underflow.

```
int top = -1; int stack[Max];
void push(int a)
{
    if (top == Max-1)
    {
        printf("\n stack is full, value can't be
                pushed overflow);
    }
    else
    {
        top = top+1;
        stack[top] = a;
    }
}
void pop()
{
    if (top == -1)
    {
        printf("\n Value can't be popped,
                as stack is empty, underflow);
    }
    else
    {
        printf("%d is deleted", stack[top]);
        top = top-1;
    }
```

```
void display() {
if (top==-1)
{   printf("\nStack is empty, nothing to display");
}
else {  int i=0;
    printf("\nStack elements: ");
    while (top!=0)
    {
        top--;
    while (i<=top)
    {   printf("%d\n", stack[i]);
        i=i+1;
    }
```

2) WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of operands & the binary operators, +, -, *, /

P1:

```
int index=0, pos=0, top=-1, length
char symbol, temp, infix[40], postfix[40],
    stack[40];
void infixtopostfix()
{
    length=strlen(infix);
    push('#');
    while (index<length)
    {   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(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;
    }
}
```

```
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=-1;
                   break;
        case '(' : p=0;
                   break;
        case '+' :
        case '-' : p=1;
                   break;
        case '*' :
        case '/' : p=2;
                   break;
    }
    return (p);
}
```

Sw
11/1/24

1) o/p :- • stack elements

1
10
100
1000

2) o/p :- Enter the infix expn : A*B+C*D-E

Postfix expn: AB*CD*+E-