

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

- a) push
- b) pop
- c) display

a) push:

```
void push(int val, int n)
```

```
{
```

```
    if (top == n-1)
```

```
        printf("overflow");
```

```
    else
```

```
    { top = top + 1;
```

```
      stack[top] = val;
```

```
}
```

```
}
```

b) pop:

```
int
```

```
void pop(int)
```

```
{
```

```
    if (top == -1)
```

```
    {
```

```
        printf("underflow");
```

```
        return 0;
```

```
    }
```

```
    else
```

```
    {
```

```
        return stack[top--];
```

```
    }
```

```
}
```

61

c) Display

```
void display()
```

```
{
```

```
    for (i = top; i >= 0; i--)
```

```
    {
```

```
        printf("%d\n", stack[i]);
```

```
    }
```

```
    if (top == -1)
```

```
    {
```

```
        printf("Stack is empty.");
```

```
    }
```

```
}
```

Q2. WAP to convert a given infix expression (valid parenthesized infix arithmetic expression) to postfix expression.

Pseudocode:

i) For operands :

```
if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') || (c >= '0' && c <= '9')) {
    result[resultIndex++] = c;
```

ii) For '('

```
else if (c == '(') {
```

```
    stack[++stackIndex] = c;
```

```
}
```

iii) else if (c == ')')

```
{
    while (stackIndex >= 0 && stack[stackIndex] != '(') {
```

```
        result[resultIndex++] = stack[stackIndex--];
```

```
}
```

```
stackIndex--;
```


iv) For '+', '-', '/', '*':

a) For precedence

```
int precedprec(char c) {
```

```
    return 3;
```

```
    else if (c == '/' || c == '*')
```

```
        return 2;
```

```
    else if (c == '+' || c == '-')
```

```
        return 1;
```

```
    else
```

```
        return -1;
```

```
}
```

b) For ~~ditam~~ operators:

```
while (stackIndex >= 0 && (preced(s[i]) < preced(stack[stackIndex]) || preced(s[i])
```

```
    == preced(stack[stackIndex])) {
```

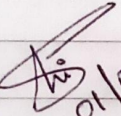
```
    result[resultIndex++] = stack[stackIndex--];
```

```
}
```

```
stack[++stackIndex] = c;
```

```
}
```

```
}
```

 01/01/2024