Priority Checker

Q) Why we need a priority checker?

***Ans*: *We need a priority checker, the reason is whether we have parenthesis* () *, or, we have* +, − ∗, /, ^, % *will act according to the priority*.**

***As we have seen that each character* ,**

+, −, %, ^, / ***as shown in table below***:

List of Operators In Expression

Symbol Used	***Operation Performed***	***Precedence***
^(***Caret***)	***Exponent*** (***Power***)	***Highest***
∗ (***asterisk***)	***Multiplication***	***Highest***
/(***Slash***)	***Division***	***Highest***
%(***percentage***)	***Modulus***(***Remainder***)	***Highest***
+(***Plus***)	***Addition***	***Lowest***
−(***hyphen***)	***Subtraction***	***Lowest***

And we have observed that each character deals with a priority even when it comes to parenthesis , the arithmetic expression must get executed in parenthesis first , hence parenthesis also carry the highest priority in infix to postfix and infix to prefix program.

Approach for the priority checker:

1) *Switch – Case statement*
2) *if – else statement*

With switch − case statement we can do:

```c
int check_prority(char c)
{
    switch(c)
    {
        case '+':
        case '-':
            return 1;
        case '*':
        case '/':
            return 2;
        case '^':
            return 3;
        case '%':
            return 4;
        case '(':
        case ')':
            return 0;
        default:
            return -1;
    }
}
```

With if − else statement we can also perform:

```c
int check_prority(char c)
{
    if (x == '+' || x == '-')
    {
        return 1;
    }
    else if (x == '*' || x == '/')
    {
        return 2;
    }
    else if (x == '^')
    {
        return 3;
    }
    else if (x == '%')
    {
        return 4;
    }
    else if (x == '(' || x == ')')
    {
        return 0;
    }
    return -1;
}
```

Note: *Here* (), ^,/,∗, %, +, − *will be pushed into the stack in infix to prefix or infix to postfix. And the character we input will be added to array. And the parenthesis will be popped without showing in the output. such as we have* $(a + b)$ *and* () *pushed into the stack and will be popped out without showing in the output.*

And priorities assigned such as +, − = 1(*lowest*), *then* ∗,/ = 2 *is higher than* + *and*− , *then* ^ *is* 3 *higher than* ∗ *and* / , *and* % *is* 4 *higher than* ^. *Then* () *will be executed at first* *when stack will be empty, hence* 0 . *Now these can be adjusted according to one’s wish*.

____________******************____________