Problem Statement:
An arithmetic expression is given in infix notation, such as:

2 + 6 * 4 / 8 - 3

We need to evaluate this expression using **stacks**.

Approach (Thought Process):

1. **Use Two Stacks:** One stack for **values** (val) and another for **operators** (op).
2. **Traverse the Expression:** Iterate through each character.
    - **If a digit is found:** Push it into the val stack.
    - **If an operator is found:** Check the precedence in the op stack.
        - If the op stack is empty or has a lower precedence operator, push the new operator.
        - Otherwise, evaluate using the top operator in op stack before pushing the new one.
3. **Evaluate the Remaining Operators at the End.**
4. **Final Result will be in val.top().**

Code Snippet:

```
#include<iostream>
#include<string>
#include<stack>
using namespace std;

int priority(char ch){
  if(ch=='+'||ch=='-') return 1;
  else if(ch=='*'||ch=='/') return 2;
  return 0;
}

int eval(int v1,int v2,char ch){
  if(ch=='+') return v1+v2;
  else if(ch == '-')return v1-v2;
```

```
  else if(ch == '*') return v1*v2;
  else return v1/v2;
}

int main(){
  string s = "2+6*4/8-3";
  stack<int> val;
  stack<char> op;
  for(int i=0;i<s.length();i++){
    if(s[i]>=48 && s[i]<=57){
      val.push(s[i]-48);
    }
    else{
      if(op.size()==0 || priority(op.top())<priority(s[i])){
        op.push(s[i]);
      }
      else{
        while(op.size()>0 && priority(op.top())>=priority(s[i])){
          int v2 = val.top(); val.pop();
          char ch = op.top(); op.pop();
          int v1 = val.top(); val.pop();
          int ans = eval(v1,v2,ch);
          val.push(ans);
        }
        op.push(s[i]);
      }
    }
  }
  while(op.size()>0){
    int v2 = val.top(); val.pop();
    char ch = op.top(); op.pop();
```

```
    int v1 = val.top(); val.pop();
    int ans = eval(v1,v2,ch);
    val.push(ans);
  }
  cout<<val.top();
  return 0;
}
```

Dry Run Table:

Step	Character	val Stack	op Stack	Action
1	'2'	[2]	[]	Push 2 into val
2	'+'	[2]	[+]	Push + into op
3	'6'	[2,6]	[+]	Push 6 into val
4	'*'	[2,6]	[+,*]	Push * into op (higher precedence)
5	'4'	[2,6,4]	[+,*]	Push 4 into val
6	'/'	[2,6,4]	[+,*]	Since * >= /, perform 6 * 4 = 24, push 24 into val
7	'8'	[2,24,8]	[+/]	Push 8 into val
8	'-'	[2,3]	[-]	Perform 24 / 8 = 3, push 3 into val
9	'3'	[2,3,3]	[-]	Push 3 into val
10	End	[2]	[]	Perform 3 - 3 = 0, Perform 2 + 0 = 2

Correct Answer: 2

Flaws in Code:

1. **Priority Function Return Issue:** If an invalid operator is encountered, the function may return garbage value. Adding return 0; fixes this.
2. **Multi-Digit Numbers Are Not Handled:** If the input has multi-digit numbers like 12+34, the code will treat 12 as 1 and 2 separately.
3. **Division by Zero Not Handled:** If v2 == 0, the program may crash.
4. **Operator Stack Overflow/Underflow Not Handled:** If operator precedence is not managed correctly, an infinite loop may occur.

Final Output:

Result: 2