C. Try and Catch

Let’s consider each block is a bracket, The “try” operator is an opened bracket “(“, the “catch” operator is a closed bracket “)” and the “throw” operator is a star “*”.

for example :

```
8
try
    try
        throw ( AE )
    catch ( BE, "BE in line 3")

    try
    catch(AE, "AE in line 5")
catch(AE,"AE somewhere")
```

will be be like this : ((*) ( ))

Now to specify the type of exception that each try/catch/throw is responsible for we will give every type of exception a unique color and every “try” / ‘(‘ , “catch” / ‘)’ and “throw” / ‘*’ operator will be colored by the color of the type of exception that is handled by it.

In the previous example let’s consider that “AE” exceptions are red and “BE” exception are green then,

it will be represented like this : ((*) ( ))

Now for general case : ((( ) ((( ) (*)) ( ))))

^ ^

we know that the exception will be handled by the last open bracket of the same color before it whose end comes after the exception, so we should print the massage mentioned in the catch operator that represented as this end closed bracket.

Consider :

int opr[] is the array of the operators
int end[]

if opr[i] is an opened bracket then, end[i] = index of it's closed bracket
otherwise end[i] = -1;

for the previous example :

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
opr	(	(	(	)	(	(	(	)	(	*	)	)	(	)	)	)	)
end	16	15	3	-1	14	11	7	-1	10	-1	-1	-1	13	-1	-1	-1	-1

Then to print the massage :

```
for ( int i = myException.position – 1 ;    i >= 0 ;    --i) {

        if ( end[i].position > myException.position && opr[i].color == myException.color ) {

                print ( end[i].msg );
                break;

        }

}
```

How to build the array end[] :

```
stack <int> myStack;

for ( int i = 0 ;    i < n ;    ++i) {

        if ( opr[i] == '(' ) {
                myStack.push ( i ) ;                    // push the index of the last "(" up to now.
        }

        else if ( opr[i] == ')' ) {

                end[i] = -1;                            // end of all )'s is -1
                int t = myStack.top ( ) ,  end[t] = i ;   // now we found the end of the last visited unclosed  "(".
                myStack.pop ( ) ;

        }

        else if ( opr[i] == '*' ) {

                end[i] = -1;                            // end of  "*"  is  -1
        }

}
```

For more about (Find index of closing bracket for a given opening bracket in an expression) :

https://www.geeksforgeeks.org/find-index-closing-bracket-given-opening-bracket-expression/

My c++ solution :

http://codeforces.com/contest/195/submission/40297482

Best Regards.