Name and ID#:___

Problem 3: Consider a text that can contain four types of braces: ( ), [ ], { } and < >. The braces are balanced, if the following two conditions hold:

1. Each time a closing brace is encountered, it matches an already encountered corresponding opening brace.
2. At the end of the text, each opening brace is matching the respective closing one.

For example, the braces are balanced in a text *{ab(c[d])e}*, but not balanced in *{ab(c))*.

Write a C++ function ***bool balanced_brackets(string text)*** or a Java method ***public static boolean balancedBrackets(String text)*** that take as the argument a string text and check, if the brackets of all four types are balanced or not. Use ***stack<char>*** in C++ or ***Stack<Character>*** in Java.

```
public static void main(String[] args){
  Scanner input = new Scanner(System.in);
  String text = input.next();
  Stack<Character> e = new Stack<Character>();
  e.pull('(');        e.pull('[');
  e.pull(')');        e.pull(']');
  e.pull('{');        e.pull('<');
  e.pull('}');        e.pull('>');
//
  balancedBrackets(e, text);

}

public static boolean balancedBrackets(Stack<Character> e, String text){
  int count = 0;
  boolean balanced = false;
  while (!e.empty()){        ← infinite loop
    int j = e.pop();
    for (int i = 0; i < text.length(); i++){
      if (text.charAt(i) == j){
        count ++;
        break
      }
  }
  if (count == 4){ balanced = true; }
  return balanced

}
```

important

4

incomplete algorithm

DS.MT1.181016.L160