Assignment: Write a program that determines if a string contains a balanced set of brackets. Brackets consist of the pairs **( )**, **[ ]**, and **{ }**. A string is a sequence of characters containing no white space. White space is a sequence of one or more blank characters, new line characters, or tab characters. A string is balanced if an opening bracket, **(**, **[**, or **{** is matched by the corresponding closing bracket, **}**, **]**, or **)**. Brackets are matched in a last-in-first-out order. If an opening curly brace, **{**, appeared in the string then the next bracket in the string must be a closing curly brace **}**. Any sequence of characters that are not brackets can appear between the opening and closing brackets.

Prohibition: **Use of the C++ Standard Template Library is prohibited in the implementation of this project. Use of template class stack presented in Lecture 10 is prohibited.**

Program Files: Project **2** consists of files **p02.cpp**, **Stack02.h**, **Stack02.cpp**, and **p02make**. Project file names are exactly as given. Failure to employ the foregoing names will result in a score of ***zero*** (***0***) for this project

Project files must be stored in the **root directory of your student account**. Failure to store project files in the root directory of your student account will result in a score of ***zero*** (***0***) for this project.

File	Description
p02.cpp	File **p02.cpp** contains functions that process command line arguments and distinguish strings having balanced brackets.
Stack02.h	File **Stack02.h** defines class *Stack*. Class *Stack* implements a character stack by dynamically allocating an array of characters. Class *Stack* is a concrete class and is not implemented via a template.
Stack02.cpp	File **Stack02.cpp** contains the implementation of member functions of class *Stack*.
p02make	File **p02make** contains instructions for the UNIX utility ***make***. Instructions in file **p02make** direct the creation of program **p02**.

Command Line: Project **2** can be invoked with zero, one, or two program parameters. The first program parameter is the input file name. The second parameter is the output file name. Sample command lines together with corresponding actions by program **p02** are shown below. Boldfaced type indicates data entered at the keyboard by the user.

$ **p02**
Enter the input file name: **i02.dat**
Enter the output file name: **o02.dat**

$ **p02 i02.dat**
Enter the output file name: **o02.dat**

$ **p02 i02.dat o02.dat**

Input File: File **i02.dat** in the class directory **~tt/cs2613/** contains a list of representative identifiers. Refer to Figure 1. Input file format.

Output File: Program **p02** produces file **o02.dat.** File **o02.dat** shown in Figure 2 is the output produced by program **p02** given the input file shown in Figure 1.

Figure 1.
Input file format

```
(({}[{()}]))
(({}[{()}]})
(o(t{b}m[s{t(f)s{t}d}f]s)p)us
(I(l{e}e[a{m(a)h{t}f}j]k}s)x
```

Figure 2.
Output file format

```
(({}[{()}])) is balanced.
(({}[{()}]}) is not balanced.
(o(t{b}m[s{t(f)s{t}d}f]s)p)us is balanced.
(I(l{e}e[a{m(a)h{t}f}j]k}s)x is not balanced.
```

Algorithm:

bool *isBalanced*(**const** *string*& *candidate*)
{ *Stack S*;
try {
for *all characters,* c_i*, of string candidate in the range* $0 \leq i < n$*, where* n *is the number of characters in string candidate* **do**
if c_i is an opening bracket, one of '(', '[', or '{' then push it on Stack S;
if c_i is a closing bracket, one of ')', ']', or '}' **then**
1. *pop character* c_j *from Stack S and*
2. *Here* c_i*is the closing bracket and* c_j *is the opening bracket. Determine if* c_j *is the corresponding opening bracket for which character* c_i *is the closing bracket. If* c_i *and* c_j *are a matched pair do nothing.* **If** c_i and c_j are mismatched **then return false,** *indicating that the candidate is not balanced.*

} catch(StackEmptyException) {
return false;
}
return true if *stack S is empty,* **false** *otherwise*;
}