Fall 2025 - CS 111 - Exam 2 Reference

111template.cpp

```
/*---
    FIRST VERSION: for an "all-in-one-file" C++ program

    COPY this into a .cpp file in a folder within the CS50 IDE

    to compile: in a CS50 terminal that is open to the folder
        CONTAINING this .cpp file, type:

        g++ your-file-name.cpp -o your-file-name

    to run: in that same CS50 terminal that is open to the folder
        CONTAINING this .cpp file, type:

        ./your-file-name

    to redirect the program's output to a .txt file:
        in the same CS50 terminal that is open to the folder
        CONTAINING this .cpp file, type:

        ./your-file-name > desired-output-file.txt

    to redirect overly-long COMPILER error messages to a
    file, compile using:

        g++ your-file-name.cpp -o your-file-name &> errors.txt

    last modified: 2025-10-16
---*/

/*---
    by:
    last modified:
---*/

#include <cstdlib>
#include <iostream>
#include <string>
#include <cmath>
using namespace std;

/*--- PUT YOUR SIGNATURES, PURPOSES, TESTS, and FUNCTION DEFINITIONS HERE ---*/

/*---
   test the functions above
---*/

int main()
{
    cout << boolalpha;

    cout << "*** Testing: put name of your function here ***" << endl;

    // put each of your function's tests into a cout statement
    //     to print its result

    // cout << () << endl;
    // cout << () << endl;

    return EXIT_SUCCESS;
}
```

Some string methods

The C++ `string` class includes a method **`length`**:

```
/*===
    method of class: string
    signature: length: void -> int
    purpose: expects nothing, and returns the number of characters in the
        calling string object
===*/
```

The C++ `string` class also includes a method **`at`**:

```
/*===
    method of class: string
    signature: at: int -> char
    purpose: expects the desired position within the calling string (where
        the position of the first character is 0), and returns the
        character at that position in the calling string as a char
===*/
```

The C++ `string` class also includes a method **`substr`**. There are actually two different `substr` methods, one that expects two arguments, and one that expects one argument. This is for the **two**-argument version:

```
/*===
    method of class: string
    signature: substr: int int -> string
    purpose: expects the starting desired position within the calling
        string (where the position of the first character is 0) and the
        length of the substring desired, and returns the substring of the
        calling string starting at that position of that length as a string
        (or until the end of the calling string if there aren't that many
        characters left)
===*/
```

Here is a description of the second **`substr`** method in the `string` class, the one that expects just **one** argument:

```
/*===
    method of class: string
    signature: substr: int -> string
    purpose: expects the starting desired position within the calling
        string (where the position of the first character is 0), and
        returns the substring of the calling string starting at that
        position and going until the end of the calling string.
===*/
```