

CS 012 • Winter 2003

In-lab Programming Exercise 1

Functions

wagner@cs.ucr.edu
Department of Computer Science and Engineering
University of California, Riverside

January 5, 2003

This first assignment is very simple and relies only on knowledge that you must have acquired in CS 010. Your task is to write a working C++ program called `in_lab_exer_1` which asks the user for three parameters — a character `c`, a positive integer `n`, and another positive integer `h` — and then prints on the console window a Christmas tree shape. For example, when you run your program and answer its prompts with the parameters `c = '+'`, `n = 4`, and `h = 2`, you should see the following printed on the console window:

```
      +
     +++
    +++++
   +++++++
  +
  +++
  +++++
 +++++++
```

The example tree above has `h = 2` branches, each with `n = 4` lines.

Your solution should be written in terms of a function with the following signature:

```
void f(const unsigned char c, const unsigned int n)
```

It's up to you to figure out what this function should do and how to use it.



Solution:

```
#include <iostream>

void f(const unsigned char c, const unsigned int n);

int main ()
{
    unsigned char c;
    std::cout << "Please enter the character to be used:\n";
    std::cin >> c;

    unsigned int n;
    std::cout << "Please enter the number of lines per branch,\n";
    std::cin >> n;

    unsigned int h;
    std::cout << "Please enter the number of tree branches,\n";
    std::cin >> h;

    // print h branches
    for (unsigned int i = 0; i < h; i++)
    {
        f(c, n);
    }

    return 0;
}
```

continued on the next page...

```

// f(c, n) prints n lines, where the top one has a single
// appearance of the character c, the second one has three
// appearances of c, the third one has five appearances of
// c, and so on, all of them centered.
//
// The key idea is to notice the patterns. If the 1st
// line has 1 character, the 2nd has 3, the 3rd has 5,
// and so on, then the n-th line will have (2*n - 1)
// characters. Also, the last line [the n-th line] has
// no blank spaces, the next-to-last [the (n-1)-th line]
// has 1 blank at the start and 1 at the end, the (n-2)-th
// line has 2 blanks at the start and 2 at the end, and so
// on. Thus, the first line has (n-1) blanks at the start
// and (n-1) blanks at the end. We can ignore the blanks
// at the end; we only need to print the blanks at the
// start and the correct number of characters.
//
// To summarize, the k-th line must have (n-k) blanks and
// then (2*k - 1) copies of the character c. We need to
// print n such lines.
void f(const unsigned char c, const unsigned int n)
{
    for (unsigned int k = 1; k <= n; k++)
    {
        // print (n-k) blanks
        for (unsigned int r = 0; r < (n - k); r++)
            std::cout << " ";

        // print (2*k - 1) c's
        for (unsigned int r = 0; r < (2*k - 1); r++)
            std::cout << c;

        // print a newline character at the end of each line
        std::cout << std::endl;
    }
}

```

■