PIC 10A: Homework 3

Michael Andrews
UCLA Mathematics Department

January 25, 2021

VERY IMPORTANT: NAMING THE FILES AS I HAVE ASKED YOU TO NAME THEM IS VERY IMPORTANT. IF YOU NAME THEM INCORRECTLY, YOU WILL RECEIVE 0 PTS.

1.
   - Write a program that reads in four `int`s (separated by spaces) and sorts them.
   - Here's a screenshot from running my code on XCode ...

     ```
     Enter four integers separated by spaces:
     4 2 3 1
     I've arranged them in order:
     1 2 3 4
     Program ended with exit code: 0
     ```

   - Submit the solution as `hw3_1.cpp`.
   - You can assume the user's input is always valid.
   - I would `#include <utility>`. Then you have the `swap` function available to you. To explain how `swap` works... The output of

     ```
     int i = 0; int j = 1; swap(i,j);
     cout << i << ' ' << j << endl;
     ```

     is `1 0`.
   - **Hints:**
     - If necessary, swap the first two `int`s so that they are in order.
     - If necessary, use more swapping to shuffle the third `int` down, so that the first three `int`s are in order.
     - If necessary, use more swapping to shuffle the fourth `int` down, so that all the `int`s are in order.

     By the end, I used `6 if` statements.
   - **Grading:** We will run your code using Visual Studio on 24 test cases and assign a score out of 4.

2. You are familiar with expressing numbers in the *decimal* system.
What do the digits of a number mean? 83526 *literally means*

$$6 \times 10^0 + 2 \times 10^1 + 5 \times 10^2 + 3 \times 10^3 + 8 \times 10^4.$$

There is nothing special about the number 10; we can use the number 2 instead. This is what computers do, and this is called the *binary* system. To express 88 in binary, we note that

$$88 = 0 \times 2^0 + 0 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 + 1 \times 2^4 + 0 \times 2^5 + 1 \times 2^6.$$

To distinguish between binary and decimal, we often use the subscripts 2 and 10, so we would write

$$88_{10} = 1011000_2.$$

Similarly, $8_{10} = 1000_2$ and $888_{10} = 1101111000_2$.

Have the first substantial lines of code you write be

```
cout << "Enter a strictly positive integer: " << endl;
unsigned int x; cin >> x;

if (x == 0) {
    cout << "This number is not STRICTLY positive" << endl;
    return 0;
}
cout << "The binary representation of " << x << " is given by ";
```

Then write more code so that your program can perform as in the example below.

```
Enter a strictly positive integer:
888
The binary representation of 888 is given by 1101111000
Program ended with exit code: 0
```

Further comments...

- Submit the solution as `hw3_2.cpp`.
- You can assume the user's input is always valid.
- **Hints:** I use two `while` loops.
  - The first calculates the largest power of `2` required.
  - The second prints each of the *bits* one by one.

  My code only makes use of one other variable: `unsigned int powerOfTwo;`
- To get started you could try to recover the usual decimal digits, and then replace `10` by `2` in your algorithm.
- **Grading:** We will run your code through Visual Studio on four test cases. In each case, a completely correct output is worth 1, and a completely nonsensical output is worth 0. You are *not* allowed to look up a function which does this question for you. Doing so will result in 0 points.

3.
- Write a program that reads in a list of integers and prints their maximum and minimum.
- You can assume that the integers in the list are separated by one space character `' '` and that the character following the last integer in the list is the newline character `'\n'`.
- Implement a loop in which the above actions are repeated until the user requests to quit.
- Here's a screenshot from running my code on XCode ...

```
Enter a list of integers:
-8 18 88
The max of the integers you entered is 88
The min of the integers you entered is -8

Continue? (y/n)
y

Enter a list of integers:
3 28 7 -2 5 11
The max of the integers you entered is 28
The min of the integers you entered is -2

Continue? (y/n)
n
Program ended with exit code: 0
```

- Submit the solution as `hw3_3.cpp`.
- **Grading:** We will run your code through Visual Studio on four test cases. In each case, a completely correct output is worth 1, and a completely nonsensical output is worth 0.

4. The *Fibonacci numbers* $F_1,\ F_2,\ F_3,\ \ldots$ are defined by the formulae:

$$\begin{aligned} F_1 &= 1, \\ F_2 &= 1, \\ F_{n+1} &= F_n + F_{n-1} \text{ for } n \geq 2. \end{aligned}$$

So the first six Fibonacci numbers are: 1, 1, 2, 3, 5, 8.

Have the first substantial lines of code you write be:

```
cout << "Type an integer bigger than or equal to 0:   " << endl;
int N; cin >> N;
cout << "The first " << N << " Fibonacci numbers are: " << endl;
```

Then write more code so that your program can perform as in the example below:

```
Type an integer bigger than or equal to 0:
8
The first 8 Fibonacci numbers are:
1 1 2 3 5 8 13 21
Program ended with exit code: 0
```

Further comments...

- Submit the solution as `hw3_4.cpp`.
- You can assume the user's input is always valid.
- **Grading:** We will run your code through Visual Studio on four test cases. In each case, a completely correct output is worth 1, and a completely nonsensical output is worth 0.