UNIVERSITY OF CALIFORNIA, MERCED

BERKELEY · DAVIS · IRVINE · LOS ANGELES · MERCED · RIVERSIDE · SAN DIEGO · SAN FRANCISCO

SANTA BARBARA · SANTA CRUZ

# CSE-165: Object-Oriented Programming
# Lab 1

Spring 2024

## Preliminary Notes

- Write a separate file for each exercise. Zip all your files together and submit the zip file to CatCourses.
- Your solution must be exclusively submitted via CatCourses. Email submissions will not be accepted. Pay attention to the posted deadline!

## 1 Reversing a file (30 points)

Modify **reversingFile.cpp** so that it opens the file **code.cpp**, reads and stores each line of the file into a vector of strings, and then prints the lines back out in reverse order with line numbers.

Expected output:

```
17: }
16:     return 0;
15:
14:     cout << (-b - (sqrt((b*b) - (4 * a * c)))) / (2 * a) << endl;
13:     cout << (-b + (sqrt((b * b) - (4 * a * c)))) / (2 * a) << endl;
12:
11:     cin >> c;
10:     cin >> b;
9:      cin >> a;
8:
7:      double a, b, c;
6: {
5: int fake_main( int argc, char *argv[] )
4:
3: using namespace std;
2:
1: #include <math.h>
0: #include <iostream>
```

## 2 Creating arrays (30 points)

Consider the **arrays.cpp** file. It calls functions declared in **createArrays.h** to dynamically allocate and print out several arrays. Your task is to create the **createArray.h** and **createArray.cpp** files and define the createArray() and printArray() functions such that they work with **arrays.cpp**.

In createArray(), the first parameter gives the size of the array; if the size is invalid (less than 1) you should print an error message and return instead. If the size is valid, dynamically allocate the array and initialize its elements with the Fibonacci sequence.

In printArray(), simply print out each element separated by a space. Remember to check if the pointer is not null.

You are not allowed to modify **arrays.cpp**.

Expected output:

```
Creating array1...
Printing array1...
0 1 1 2 3

Creating array2...
ERROR: size of array < 1
Printing array2...
Invalid array.

Creating array3...
Printing array3...
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

## 3 Bit manipulation (40 points)

a) Write a program that reads in a decimal number "n" and a position "index". Convert the decimal number into binary format and print out the number in binary format.

b) Next, implement the following functions to get, set, and clear the bit at position "index" of the number.

*/* Retrieve a bit from a number "n" in binary format at position "index"*
*Input: number n, position index with 0 being the right most (least significant) bit*
*Output: bit at position "index"*
*Example: Input: n=1010, index=1, output=1 */*
**int getBit(int n, int index)**

*/* Set a bit at position "index"*
*Input: number n, position index with 0 being the right most (least significant) bit*
*Output: the binary number after the bit is set at position "index"*
*Example: Input: n=1010, index=0, output=1011 */*
**int setBit(int n, int index)**

*/* Clear a bit at position "index"*
*Input: number n, position index with 0 being the right most (least significant) bit*
*Output: the binary number after the bit is cleared at position "index"*
*Example: Input: n=1010, index=1, output=1000 */*
**int clearBit(int n, int index)**

c) Finally, using your functions, print the following three things: 1) the original bit at position "index"

of “n”; 2) the number “n” with the bit at “index” set; 3) the number “n” with the bit at “index” cleared. Essentially, use the two inputs to the program to call your three functions and print their returns.

**Example Inputs:** 10 and 2
**Output:**
Binary representation of 10 is 1010
Get bit at index 2: 0
Binary number after setting bit at index 2: 1110
Binary number after clearing bit at index 2: 1010

**Example Inputs:** 547 and 8
**Output:**
Binary representation of 547 is 1000100011
Get bit at index 8: 0
Binary number after setting bit at index 8: 1100100011
Binary number after clearing bit at index 8: 1000100011

**Example Inputs:** 1023 and 5
**Output:**
Binary representation of 1023 is 1111111111
Get bit at index 5: 1
Binary number after setting bit at index 5: 1111111111
Binary number after clearing bit at index 5: 1111011111