EECE 231 — Introduction to Programming Using C++ and MATLAB
Programming Assignment 3

September 24, 2014

- This programming assignment consists of 4 problems.
- It is due at the beginning of the next lab session by 3:00 pm. You are required to submit your solutions of the lab assignment during the the lab time. The instructor will supervise the submissions and randomly select students and check their understanding of the submission to control cheating. You are supposed to submit your own work. Cheating will not be tolerated and will be penalized (zero grade on the programming assignment, disciplinary committee, faling course grade).
- Related reading: Selection, Repetition, and Arrays.
- *Lab structure and regulations:*
  - ⋆ The lab consists of:
    - *Occasional Solving Session (not graded but attendance mandatory)*
    - *Programming Assignment (graded)*
      Programming Assignments will be posted on Moodle on weekly basis. Typically, a Programming Assignment requires much more than the time allocated for this part in the Lab, so you are supposed to complete a part of the assignment at home. The Lab instructor will grade your assignment and help you with the problems you are facing.
    - *Occasional graded weekly quiz*
  - ⋆ Lab attendance is mandatory.

Problem 1. Pattern search

a) Write a program which reads a sequence of zeros and ones (separated by spaces) and checks whether or not the sequence contains the pattern 0 0 1. Thus your program is supposed to give a YES/NO answer.

For example, each of the following sequences contains the pattern 0 0 1 (underlined).

1 0 0 1 1 0 1
1 0 1 0 0 0 0 0 0 0 1 0 1
0 0 0 0 1 1 0 0 1 1 0 1

On the other hand, none of the following sequences contains the pattern 0 0 1.

0 1 0 1 0 1 0 1
1 0 1 1 1 1 1 1 0 1 0 1

Your program should ask the user to enters first the number of entries in the sequence.

It is easier to solve this problem using an array. You are asked to solve this part by first storing the sequence in an array, and then processing the array.

b) **(Optional)** Repeat part (a) without using arrays.

Problem 2. Checking if a binary sequence is symmetric

A sequence of zeros and ones is called symmetric if it reads the same both forward and backward. For example, the sequences

```
1 0 1 1 0 1
1 0 1
0 1 1 0
0 1 0 1 0
1 0 1 1 0 1 1 0 1
1
```

are all symmetric, but

```
1 1 1 0
0 0 1 0 1
```

are not symmetric.

Write a program which reads a sequence of zeros and ones (separated by spaces) and checks whether or not the sequence is symmetric.

Thus your program is supposed to give a YES/NO answer.

Your program should ask the user to enters first the number of integers in the sequence.

Do not read the following hint unless you are stuck. Think first.

(*Hint:* To solve this problem, you have to store the sequence in an array $A[0 \ldots n-1]$ of type *bool*. You need a boolean variable initialized to *true*. Traverse the array using a loop: compare $A[0]$ with $A[n-1]$, then $A[1]$ with $A[n-2]$, and so on ... . When you see a violation, set the boolean variable to false, and ... . Do not print anything in the body of the loop: postpone printing "symmetric" or "not symmetric" till the end.)

Problem 3. Element distinctness problem

Write a program which prompts the user to enter an integer n, and list of n integers. Your program is supposed to check whether or not all the n integers in the list are distinct.

For instance, the list $30, 7, 9, 7, 10$ does not consist of distinct integers since 7 appears twice.

Your program is supposed to give a YES/NO answer only.

(*Hint:* Use an array and nested loops).

Problem 4. Squares (more on loops and boolean variables)

a) **Square test.** Write a program which prompts the user to enter an integer n, and checks whether or not n is a square of another integer. That is, your program should check whether or not there exists an integer x such that $n = x^2$.

For instance, the following are squares: 0 (since $0 = 0^2$), 1 (since $1 = 1^2$), 4 (since $4 = 2^2$), 25 (since $25 = 5^2$), and 81 (since $81 = 9^2$). On the other hand, -25, 2, 3, 10 are not squares.

If the answer is YES, your program is supposed to print x in the format shown below.

You are *NOT allowed* to use the power or the square-root functions or any other function in the *cmath* library.

As in the primality test problem, use a boolean variable $isSquare$ initialized to $false$. Check if such an x exists using a loop by first trying $x = 0$, then $x = 1$, then $x = 2$, and so on ... Stop the loop when x^2 is equal to y or x^2 is larger than y, and appropriately modify the boolean variable in the loop.

Format:

```
Please enter an  integer: 3
3 is not a square

Please enter an  integer: 81
YES square: 81=9^2
```

b) **Sum of two squares.** Write a program which prompts the user to enter an integer n, and checks whether or not n is the sum of two squares. That is, your program should check whether or not there exist two integer x and y such that $n = x^2 + y^2$. If the answer is YES, your program is supposed to print x and y in the format shown below.

As in (a), you are *NOT allowed* to use the power or the square-root functions or any other function in the *cmath* library.

For instance, each of the following number is a sum of two squares: 0 (since $0 = 0^2 + 0^2$), 1 (since $1 = 0^2 + 1^2$), 2 (since $1 = 1^2 + 1^2$), 4 (since $4 = 0^2 + 2^2$), 5 (since $5 = 1^2 + 2^2$), 8 (since $8 = 2^2 + 2^2$), 9 (since $9 = 0^2 + 3^2$), and 10 (since $10 = 1^2 + 3^2$). On the other hand, none of the following numbers is a sum of two squares: -10, 3, 6, 7, 11.

Format:

```
Please enter an  integer : 5
YES: 5 = 1^2 + 2^2

Please enter an  integer: 6
6 is not a sum of two squares
```

(*Hint:* Use two nested loops and a boolean variable.)