EECE 230 Introduction to Programming, Sections 3,4, and 12
Programming Assignment 4

Tuesday Oct 16, 2012

- This programming assignment consists of 4 problems.
- It is due on Tuesday Oct 23 in the lab.
- Related reading: Control Structures, Arrays, Insertion Sort.
  We will cover Insertion Sort on Thursday October 20.
- *Lab structure and regulations:*
  - ⋆ The 3 hours Lab session is on Tuesdays in Lab rooms 1,2, and 5 from 2:00 pm to 5:00 pm. It consists of three parts:
    - *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 the major part of the assignment at home. The Lab instructor will grade your assignment and can help you with the problems you are facing.
    - *Occasional graded weekly quiz*
  - ⋆ You are supposed to submit your own work. Cheating will not be tolerated and will be dealt with severely: zero grades on the programming assignments, disciplinary committee, Dean's warning.
  - ⋆ Lab attendance is mandatory. Violating this rule can lead to a failing grade.

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:* .dne eht llit "cirtemmys ton" ro "cirtemmys" gnitnirp enoptsop :pool eht fo ydob eht ni gnihtyna tnirp ton oD . ... dna ,eslaf ot elbairav naeloob eht tes ,noitaloiv a ees uoy nehW. ... no os dna $,[2-n]A$ htiw $[1]A$ neht $,[1-n]A$ htiw $[0]A$ erapmoc :pool a gnisu yarra eht esrevarT *.eurt* ot dezilaitini elbairav naeloob a deen uoY $.[1-n\ldots 0]A$ yarra na ni ecneuqes eht erots ot evah uoy ,melborp siht evlos oT)

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. Selection Sort.

We studied in class the Insertion Sort algorithm and its C++ implementation. You will design and implement in this problem another algorithm for the sorting problem called Selection Sort. The idea of Selection of Sort is the following:

To sort an array $A[0...n-1]$:

1. find the index $minIndex$ of smallest element in $A[0\ldots n-1]$
2. exchange $A[minIndex]$ (swap it) with $A[0]$, hence now $A[0]$ is the smallest element of $A[0\ldots n-1]$, i.e., the number stored in $A[0]$ is in its correct position in the desired sorted order.
3. find the index $minIndex$ of the smallest element in $A[1\ldots n-1]$
4. exchange $A[minIndex]$ with $A[1]$, hence now $A[1]$ is the second smallest element of $A[0\ldots n-1]$, i.e., the number stored in $A[1]$ is in its correct position in the desired sorted order.

5. find the index $minIndex$ of the smallest element in $A[2 \ldots n-1]$
6. exchange $A[minIndex]$ with $A[2]$, hence now $A[2]$ is the third smallest element of $A[0 \ldots n-1]$, i.e., the number stored in $A[2]$ is in its correct position in the desired sorted order.
7. and so on until $A[0 \ldots n-1]$ is sorted

The objective of this problem is to design and implement this algorithm.

a) Illustrate the algorithm on the following example

5 2 4 6 1 3

b) Write a pseudocode for Selection Sort.

You need two nested loops, where the inner loop is used to find the index smallest element in subarray.

(*Note*: Say that we want to exchange (i.e., swap) $A[7]$ and $A[2]$. If you set $A[7]$ to $A[2]$, and then set $A[2]$ to $A[7]$, you loose the old value of $A[7]$. To resolve this issue, use a temporary variable *temp*:

1. set a *temp* to $A[7]$
2. set $A[7]$ to $A[2]$, then
3. set $A[2]$ to *temp*.)

c) Implement Selection Sort in C++. Your computer program should take as input a sequence of integers whose end is indicated by a sentinel.

(Note: In addition to the C++ code in part (c), you are supposed to show your work on parts (a) and (b) to the Lab instructor).