EECE 230 Introduction to Programming, Sections 3,4, and 12
Programming Assignment 7

Tue Nov 13, 2012

- This programming assignment consists of 4 problems.
- It is due on Tuesday Nov 20 in the lab.
- Related reading: passing parameters to functions, binary search, merge function.
- *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. (Pointers and reference parameters practice)
Write the functions:

a) *divide*

This function takes as input parameters two integers a and b. It returns via reference parameters the quotient q and the remainder r resulting from the division of a by b (i.e., q and r are nonnegative integers such that $a = qb + r$ and $r < b$).

b) $dividePtrsVersion$

Same as (a) but use pointers instead of reference parameters.

c) $rotateThree$

This function takes three integer variables by reference.

It is supposed to rotate the content of the variables.

For example, the following test program

```
int a = 2;
int b = 5;
int c = 7;
rotateThree(a,b,c);
cout<<a<<'' ''<<b<<'' ''<<c;
```

should output

```
7 2 5
```

d) *rotateThreePtrsVersion*

Same as (c) but use pointers instead of reference parameters.

e) *fileStatistics* (revisited)

This functions takes 4 parameters: the name of a text file (a C-string), and 3 parameters by reference. It is supposed to return via the the 3 reference parameters: the number of characters, the number of words, and the number of lines in the file [Problem 1, Programming Assignment 4].

All what you have to do is to use the code from the solution of Problem 1 of PA 4 and appropriately put it in the body of a function.

Write a program to test your functions.

Problem 2. Binary Search base case modified

First review the sequential search and the binary search algorithms (notes on Moodle).

The version of binary search we studied in class keeps on dividing the array in halves until either we find the element we are searching for or we reach an empty array when the element we are after is not in the array.

Modify the base case of binary search by performing a sequential search when the subarray size becomes less than or equal to 5. That is, we keep on dividing in halves until either we find the element we are searching for or we get a subarray of size less than or equal to 5, in which case we perform a sequential search.

Implement the function

```
int binarySearch5(const int  A[],int n, int x)
```

The function is supposed to return the index of x in A if found and -1 otherwise.

Write a program to test your function.

Test your function on large arrays. Test it on the array of even number $2, 4, 8, 10, \ldots, 198, 200$ and use the function to search for the numbers -2, 2, 13, 20, 56, 100, 157, 180, 183, 199, 200, 250.

Problem 3. Fast sorted 2-sum

Recall the 2-sum problem from Solving Session 2 and consider the special case when the list of numbers is sorted.

Sorted 2-sum problem: Given a list of n integers *sorted* in nondecreasing order and target integer t, checks whether or not the list contains two integers x and y whose sum is equal to t. If the answer is YES, we want to find any such pair (x, y) of integers (not all).

In PSS2, we solved this problem in quadratic time. In this problem, we will take advantage of the fact that the array is sorted to solve it more efficiently. We will use two different approaches.

a) Hint: the idea is to **use** the **binary search** function n times ...

($x - t$ rof hcraes, yarra eht fo x tnemele hcae roF)

Call your function

```
bool  fastSorted2SumA(const int A[], int n, int t, int & x, int & y);
```

Thus x and y if found are returned by reference.

b) **(Optional)**

Hint: the idea is to use two counters: i initialized to 0 and j to $n-1$ .... try to do something **similar to the merge function**. Compare $A[i]+A[j]$ to t, and accordingly decide how to move i and j.

The objective is to achieve linear time. Hence (b) should be faster than (a).

Call your function

```
bool  fastSorted2SumB(const int A[], int n, int t, int & x, int & y);
```

Write a program to test your functions. Note that they are not supposed to work properly if the array is not sorted.

Problem 4. Partition

Consider the Partition Problem:

We are given an array $A[0 \dots n-1]$ and its size n.

Let x be the last element in the array, i.e., $x = A[n-1]$.

This problem is about partitioning A around x. More precisely, you are supposed to rearrange A in a such a way that:

1. Each element of $A[0 \dots q-1]$ is less than or equal to x
2. $A[q] = x$
3. Each element of $A[q+1 \dots n-1]$ is greater than or equal to x

Thus q is the new index of x in the modified array.

In addition to rearranging the array, you are supposed return q.

Note that we don't care about the order of the elements within $A[0 \dots q-1]$ and within $A[q+1 \dots n-1]$. We only care about their relation to x as explained in (1),(2), and (3) above.

Thus, typically, there is no unique solution of the Partition problem.

For instance if

$$A = \{1, 3, 2, 8, 5, 6, 12, 4, 11, 9, 7\},$$

hence $x = 7$, a valid solution of the partition problem is the following reordering of A

$$A = \{1, 3, 2, 4, 5, 6, 7, 8, 11, 9, 12\}.$$

Thus $q = 6$. Note that

$$1, 3, 2, 4, 5, 6 < 7 < 8, 11, 9, 12.$$

Write a function for the Partition Problem and write a program to test it.

Note that you are not supposed to sort the array. You are asked to partition the array in a more efficient way.

You are not allowed to use temporary arrays. Suitably traverse the array and successively apply the the swap function when needed.

There is more than one algorithm for this problem. Here is the idea of one of them. Do not read the following hint unless you are stuck. Think first.

(*Hint:* .noitisop thgir sti ni $[1-n]A$ ecalp ew dne eht tA .tuo erugif ot evah uoy taht snoitidnoc yradnuob emos hcaer ew litnu ssecorp eht taeper ew dna ,$[j]A$ dna $[i]A$ egnahcxe ew tniop siht tA .$[j]A \geq x$ noitidnoc eht fo noitaloiv tsrif eht hcaer ew litnu j gnitnemerced no peek ew dna ,$x \geq [i]A$ noitidnoc eht fo noitaloiv tsrif eht hcaer ew litnu i gnitnemercni no peek ew nehT .$2 - n = j$ dna $0 = i$ sretnuoc owt ezilaitini eW)

Before writing the C++ code of the function, elaborate on the idea, try it on on some examples, and write a pseudocode.