EECE 230 Introduction to Programming, Sections 3 and 4
Quiz II

Dec 22, 2009

- The duration of this exam is 2 hours and 45 minutes.
- It consists of 4 problems.
- The exam is open book. You can use also all the material on Moodle: lecture notes, programming assignments, and solutions, etc. You are **NO**T allowed to use the **web** (**imail** included). You are not allowed to use **USB's** or files previously stored in your **account**.
- If you violate the above rules or if you communicate with a person other than the exam proctors during the exam, you will immediately get zero and you will be referred to the appropriate disciplinary committee.
- Active cell phones and any other unauthorized electronic devices are absolutely not allowed in the exam rooms. They should be turned off and put away.
- Plan your time wisely. Do not spend too much time on any one problem. Read through all of them first and attack them in the order that allows you to make the most progress.
- Submit your solutions each part in a separate file as indicated in the booklet. Include your name and ID number in each file. Submit the files online in a single zip file called *yourLastName.yourFirstName.zip*.
- Good luck!

Problem 1 (25 points). Files

Write a program which prompts the user to enter a C-string containing the name of an input text file. Your program is supposed to check whether or not the 10'th character in the file is the character 'a'. It is supposed to detect also the case in which the file contains less than 10 characters. Thus your program is supposed to print one of the following messages: "*The 10'th character is 'a'.*", "*The 10'th character is not 'a'.*", or "*The file contains less than 10 characters.* ".

Note that white spaces are included in the character count.

Submit your code in a file called Prob1.cpp. Include your name and ID number in the file.

Problem 2 (30 points). Strings

Write a function *alternatingDelete*, which given a *C*-string *s* as input argument, modifies *s* by deleting every other character of *s* starting with the second character.

Examples:

- If $s =$ "Problem 2", then after calling the function *alternatingDelete* on *s*, *s* becomes "Polm2".
- If $s =$ "aabc", then after calling the function *alternatingDelete* on *s*, *s* becomes "ab".

Use the following test program

```
#include<iostream>
using namespace std;

// alternatingDelete prototype:
...
int main()
{
     char s[500];
     cout << ‘‘Enter string:’’;
     cin.get(s,500);// read a line (possibly containing spaces) of length
                    // at most 499 characters and strote it into s.

     alternatingDelete(s);
     cout << "Every other chracter deleted:"<<endl;
     cout<< s<<endl;
     return 0;
}
// alternatingDelete body:
...
```

To get full credit, don't use a temporary array.

Submit your code in a file called Prob2.cpp. Include your name and ID number in the file.

Problem 3 (30 points). Recursion: count the even numbers

In this problem, we are given:

1. an integer n
2. a pointer to a size-n array of integers A consisting of a list of even numbers *followed by* a list of odd numbers.

We are asked to find the number of even numbers in A. Examples:

- for $n = 10$ and $A = \langle 2, 8, 4, 16, 2, 2, 100, 3, 11, 5 \rangle$, the number of even numbers is 7
- for $n = 4$ and $A = \langle 2, 10, 4, 82 \rangle$, the number of even numbers is 4.
- for $n = 4$ and $A = \langle 7, 3, 5, 1 \rangle$, the number of even numbers is 0.

a) **(10 points)** Write a function $numberOfEven$ which solves the problem using a for-loop (scan the array starting from the beginning and return the number of even numbers when the first odd number is found).

b) **(20 points)** Write a **recursive** function $fastNumberOfEven$ which solves the problem more efficiently by imitating binary search: look at the middle element of the array and ... proceed recursively on the left subarray or the right subarray.

Prototype:

```
int  fastNumberOfEven(int  *A, int start, int end);
```

Initial call: $start = 0$ and $end = n - 1$.

Note that your function is not required to work properly if the array is not in the correct format (i.e., if it is not a list of even numbers followed by a list odd numbers).

Write a program to test your functions.
Submit your code in a file called Prob3.cpp. Include your name and ID number in the file.

Problem 4 (15 points + 10 bonus points). Bounded print all recursively

Let m, n, and sum be positive integers. In this problem we are interested in length-n arrays whose entries are integers in the set $\{1, 2, \ldots, m\}$ such that the sum of the entries of the array is less than or equal to sum.

Write a recursive function, which given m, n and sum, prints all such arrays. Your function should also compute the total number N of those arrays.

Examples:

- For $m = 2, n = 3$, and $sum = 4$, the function should print

```
1 1 1
1 1 2
1 2 1
2 1 1
```

Thus $N = 4$.

- For $m = 3, n = 3$, and $sum = 5$, the function should print

```
1 1 1
1 1 2
1 1 3
1 2 1
1 2 2
1 3 1
2 1 1
2 1 2
2 2 1
3 1 1
```

Thus $N = 10$.

- For $m = 3, n = 4$, and $sum = 6$, the function should print

```
1 1 1 1
1 1 1 2
1 1 1 3
1 1 2 1
1 1 2 2
1 1 3 1
1 2 1 1
1 2 1 2
1 2 2 1
1 3 1 1
2 1 1 1
2 1 1 2
2 1 2 1
2 2 1 1
3 1 1 1
```

  Thus $N = 15$.

The time of your function should be in the order of the output size N (and not m^n). That is, you are expected to avoid recursive branches which do not eventually lead to valid arrays.

Submit your solution in a file called Prob4.cpp including your name and ID number.