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

Dec 20, 2011

- The duration of this exam is 2 hours and 45 minutes.
- It consists of 4 problems. The total number of points is 120.
- The exam is open moodle. You can use all the material on Moodle: lecture notes, programming assignments, and solutions, etc. You are **NOT** 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 $abcMN.zip$, where $abcMN$ is your AUB user i.d., e.g., $abc01.zip$.
- Good luck!

Problem 1 (20 points). Files

Write a program which creates a new file called "square.txt", prompts the user to an enter an integer n, and stores in "square.txt" an $n \times n$ square of the form:

- If $n = 1$

  ```
  *
  ```

- If $n = 2$

  ```
  **
  **
  ```

- If $n = 3$

  ```
  ***
  * *
  ***
  ```

- If $n = 4$

  ```
  ****
  *  *
  *  *
  ****
  ```

- ...

If the user input n is negative or zero, your program is supposed to display an error message "n less than or equal to zero!" and keep the file "square.txt" empty.

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

Problem 2 (50 points+10 bonus points). Duplicated Strings

a) **(20 points) Check if duplicated.** Write a function *checkIfDuplicated*, which given a *C*-string s as input argument, checks if s is the concatenation of two copies of the same string. *Examples:*

   - "abcabc" is the concatenation of two copies of the same string ("abc").
   - "zuc1zuc1" is the concatenation of two copies of the same string ("zuc1").
   - "qq" is the concatenation of two copies of the same string ("q").
   - "abcab" is NOT the concatenation of two copies of the same string.
   - "abab2" is NOT the concatenation of two copies of the same string.
   - "x" is NOT the concatenation of two copies of the same string.

   Your function is supposed to return a boolean value (*ture* if the answer is YES and $false$ if the answer is NO). Use the following test program:

```
... checkIfDuplicated prototype
int main()
{
     char s[100];
     cout<<"Enter string:"<<endl;
     cin>>s;
     ... call the function checkIfDuplicatded on s and depending on
     ... its return value  display "YES duplicated" or "NO not duplicated"
     cout<<endl;
     return 0;
}
... checkIfDuplicated function body
```

For simplicity, assume that the user input does not contain white spaces. Test your function on the above examples.

Submit your code in a file called Prob2a.cpp including your name and ID number.

b) **(30 points + 10 bonus points) Longest duplicated substring.**

Write a function which given a C-string s, finds the longest duplicated substring of s, i.e., the longest substring of s which appears twice in s (without overlapping with itself).

Examples':

1. If $s =''$ $xyzabcdxyabcd12''$, then the longest duplicated substring is $''abcd''$ as it appears twice without overlapping with itself and it is the longest substring of s with this property.
2. If $s =''$ $aaaaa''$, then the longest duplicated substring is $''aa''$. It appears twice without overlapping with itself: "**aa**aaa'' and "aa**aa**a'' (in bold). Note that "$aaaa''$ is not a valid answer since the substrings "**aaaa**a'' and "a**aaaa**$''$ (in bold) overlap.
3. If $s =''$ $123xyxy312u3''$, then a longest duplicated substring of $''12''$. Another valid answer is $''xy''$ since it also appears twice and it has the same length as $''12''$.
4. If $s =''$ $abcxyxa''$, then the longest duplicated substring is $''x''$.
5. If $s =''$ xyz''', then a longest duplicated substring is the empty string.

Note that the longest duplicated substring is not necessarily unique (e.g., Example 3 above). We are satisfied with any of the longest ones.

Use the following test program and try the above examples.

```
void longestDuplicatedSubstring(char s[],char duplicated[]);
int main()
{
     char s[100];
     cout<<"Enter string:"<<endl;
     cin>>s;
     char duplicated[50];
     longestDuplicatedSubstring(s,duplicated);
     cout<<"Longest duplicated substring:"<< duplicated<<endl;
     return 0;
}
void longestDuplicatedSubstring(char s[], char duplicated[])
{
     .....
}
```

Any correct solution is worth 20/30 points. If you solve without triply-nested loops, you get 40/30 points

Submit your code in a file called Prob2b.cpp including your name and ID number.

Problem 3 (30 points). Recursive odd numbers counter

a) **(10 points) Simple for loop.**

Write a function *oddNumbersCounter*, which given an array A of integers and its length n, returns the number of odd integers in A.

Examples:

- If $A[] = \{10, 5, 12, 1, 3, 4, 8, 11\}$, the function is supposed to return 4 (since A contains 4 odd numbers: $5, 1, 3, 11$).
- If $A[] = \{12, 4, 2, 20, 6\}$, the function is supposed to return 0 (since all the numbers in A are even).
- If $A[] = \{11, 3, 1, 19, 5\}$, the function is supposed to return 5 (since all the numbers in A are odd).

Use the following test program which is based on the above examples.

```
...oddNumbersCounter function prototype
int  main()
{

 int A[] = {10,5,12,1,3,4,8,11};
 ...call oddNumbersCounter on array A and display its answer


 int B[] = {12,4,2,20,6};
 ...call oddNumbersCounter on array  B and display its answer

 int C[] = {11,3,1,19,5};
...call oddNumbersCounter on array C and display its answer

 return 0;
}
...function  oddNumbersCounter body
```

Submit your code in a file called Prob3a.cpp including your name and ID number.

b) **(20 points) Recursive function.**

Solve Part (a) using a recursive function. The use of **for** or **while** loops in the recursive function is strictly **prohibited**. You are asked to use recursion instead. A solution based on for or while loops is worth zero points. Call your function *recursiveOddNumbersCounter*. Prototype:

```
int recursiveOddNumbersCounter(int A[], int a, int b);
```

where a is the start and b is the end. Thus the initial call is

```
cout<<recursiveOddNumbersCounter(A,0,n-1);
```

where n is the length of A.

Write a test program and use the above examples.

Submit your code in a file called Prob3b.cpp including your name and ID number.

Problem 4 (20 points). Print all sorted arrays recursively

Let m and n be positive integers. In this problem we are interested in *sorted* length-n arrays whose entries are integers in the set $\{1, 2, \ldots, m\}$. We are interested in such arrays which are sorted in *nondecreasing order*.

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

Examples:

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

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

  Thus $N = 3$.

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

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

  Thus $N = 4$.

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

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

  Thus $N = 21$.

- For $m = 5$ and $n = 2$, the function should print

```
1 1
1 2
1 3
1 4
1 5
2 2
2 3
2 4
2 5
3 3
3 4
3 5
4 4
4 5
5 5
```

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 sorted array.

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