Introduction To Computer – C (234112)

Spring 2013\Summer 2013

Moed B

Duration: 119 minutes. You are not allowed to exit the classroom to use the bathroom. 
External Material: you are not allowed to use any written, printed or electronic external material.

Guidelines and Instructions:
- Write your answers only on the exam form, in the intended places. Note that the given place doesn't necessarily indicate the length of the correct answer.
- The even pages of the exam form are empty. You can use them as a draft or to write your answers. Write drafts clearly, so they won't be checked.
- Write your answers tidy, clean and clearly as possible. You're allowed to use pencil and an eraser, but you must fill the title page in pen.
- In all of the questions, you're allowed to define (and implement) your own functions.
- You are not allowed to use global and/or static variables, or pre-compilation commands (include\define).
- You are not allowed to use library functions, or functions implemented in class, without implementing them yourself, unless noted explicitly in the question, excluding input\output functions.
- In every question, you are allowed to use functions defined in previous parts of the same question, even if you didn’t solve these parts, though this is not obligated.
- You don't need to check input correction, unless explicitly noted in the question.
- You don’t need to implement the main function, and the order of writing the functions is not important.
- the complexity of the solution will not be marked, unless explicitly stated.
- It's recommended to add a written explanation of your algorithm. That explanation will not be marked.
Question 1 (25 points)
An integer is called extremely even if the sum of its digits is even. 
Example: 101, -101, 1234 are extremely even. 12, -15, 12345 are not extremely even

Part 1 (10 points)
Write the following function:

int veryEven(int n)

Its parameters are an integer, n. the function will return 1 if n is extremely even, and 0 (zero) if n is not extremely even.

int veryEven(int n){
}
A Magnificent Pair in an array of integers, int a[N] are two adjacent numbers in an array (a[i],a[i+1]), one of them is extremely even, and the other one is not extremely even.

Example: for the following array:

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>33</td>
<td>22</td>
<td>17</td>
<td>16</td>
<td>27</td>
<td>31</td>
<td>32</td>
</tr>
</tbody>
</table>

The magnificent pairs are 17,16 (in indexes 3,4), 27,31 (in indexes 3,4), 31,32 (in indexes 6,7).

Part 2 (15 points)
Write the following function:

```c
int findAmazing(int a[], int n)
```

Its parameters are an array of integers, a. Its first element, a[0], is extremely even, and its last element, a[n-1], is not extremely even. The function returns an index i in the array, where a[i], a[i+1] are a magnificent pair.

Note – the implementation of the search needs to be efficient, i.e. a serial search in which you scan the array, or part of it, an element at a time, will lose points.

```c
int findAmazing(int a[], int n){
    // Your implementation here
}
```
Question 2 (20 points)

Write the following function:

```c
int substring(char *str, char *substr)
```

its parameters are two strings. The function returns the number of times the second string appears in the first string.

**Example:** `substring("abcdbxyzbcdbcdxyz", bcdb")` will return 3.

No special efficiency requirements, as long as you don't do completely unnecessary things.

```c
int substring(char *str, char *substr){

```

Question 3 (30 points)

In this question we will sort strings of characters by their order in the ASCII table.

Part 1 (15 points)

Write the following function:

```c
void sortR(char *s)
```

Its parameter is a string of characters. The function sorts the string, following these guidelines:

- The entire implementation needs to be based off the "bubble sort" algorithm.
- **The entire implementation needs to be recursive.**
- You are **not allowed to use loops**
- You are not allowed to use library functions, but you can implement and use your own functions (in their implementation you're not allowed to use loops as well).
- You can implement and use a function similar to `strlen()` (without using loops), but a solution that doesn't calculate the length of the string will get a bonus of 4 points.

```c
void sortR(char *s){
```

```c
}
```
Part 2 (15 points)
Write the following function:

```c
void sortLinear(char *s)
```

Its parameter is a string of characters. The function sorts the string, following these guidelines:
- You're allowed to traverse the string a constant (small, not dependent on the length of the string) number of times!
- You're allowed to use loops.
- You're allowed to use an extra array with a constant, predefined size, which should not depend on the length of the string.

```c
void sortLinear(char *s){
```

```c
}
```
Question 4 (25 points)

Write the following function:
```c
int colRow(int a[][N])
```

Its parameter is a square matrix of size \( N \times N \) (\( N \) is defined in a define statement). The function returns the value \( i \), if there’s an \( i \)'th row and an \( i \)'th column that are completely identical, when you compare the elements of the \( i \)'th column from top to bottom, and the elements of the \( i \)'th row from right to left. If there are no such row and column, the function will return -1 (minus 1).

Example: for this matrix, the function will return 3.
```
5 0 7 1 6
2 1 9 2 8
3 2 5 3 4
5 2 3 2 1
6 6 6 5 6
```

And for this matrix, the function will return -1.
```
5 0 7 1 6
2 1 9 2 8
3 2 5 3 4
5 4 3 2 1
6 6 6 5 6
```

```c
int colRow(int a[][N])
```