Introduction To Computer – C (234112)

Winter 2011

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.
• It's recommended to add a written explanation of your algorithm. That explanation will not be marked.
Question 1 (35 Points)
Part 1 (15 Points)
Write the following function:

```c
int is_prime(int n)
```

its parameter is a number, n. the function checks whether n is prime or not, and returns 1 if it is prime, and 0 otherwise.

Reminder – a prime number is a natural number (whole and positive), greater than one, that can't be presented as a product of two natural numbers that are lower than itself.
The efficiency of the solution will not be checked.
Part 2 (20 Points)

In this part we will check Goldbach's Conjecture, which says every even number greater than 2 can be written as a sum of two prime numbers. Write the following function:

```c
Int find_primes(int n, int *p1, int *p2)
```

It's parameters is a whole, even number, `n`, which is greater than 2. If `n` can be presented as two prime numbers, the function returns 1. Otherwise, it returns 0. If `n` can be presented as two prime numbers, the function will return the numbers with the pointers `p1` and `p2`. If there are several solutions, the function can return any of them.

**Efficiency of the solution will not be checked.**

You can assume `p1` and `p2` point to a legal address in the memory.
Question 2 (30 Points)
Part 1 (15 Points)
Write the following function:

```c
int does_generate(char *s, char *sub, int len)
```

its parameters are a legal string, s, an array of chars, sub, and it's length, len. The function will return 1 if a concatenation of the char array sub makes the string s. the function will return 0 otherwise.

Examples:
1. For s="abcabcabcabc", sub=abc, len=3, the function will return 1, because concatenating abc for times makes the string s.
2. For s="abcabcabcab", sub=abc, len=3, the function will return 0, because no concatenation of abc will make the string s.

You can assume the input is only chars and numbers.

Remainder – you are not allowed to use library functions without implementing them.
Part 2 (15 Points)

A "preface" of a string is an array of characters in the beginning of a string. An example – the string "abca" has four different "prefaces", which are:

- a – the first character
- ab – the first two characters
- abc – the first three characters
- abca – the first four characters

Write the following function:

```c
int shortest_str(char *s)
```

Its parameters are a legal, non-empty string s. the function returns the length of the shortest "preface", which concatenation makes the string s. (for more details, look at Part 1 of this question). You can assume the input is only chars and numbers.

**Remainder – you are not allowed to use library functions without implementing them.**

**Examples:**

1. For s="abcabc", the function will return 3, because abc is the shortest preface that makes the string s.
2. For s="abcab", the function will return 5, because abcab is the shortest preface that makes the string s.

**What is the time complexity of the solution you’ve written (including Part 1 if you’ve used it)? _____ (3 points)**
**Question 3 (35 Points)**

The **Stability** of an array is defined as the difference between the sum of the first half of its elements and the sum of the second half of its elements. In an array with an odd number of elements, the middle element is included in both halves.

Example 1: for the array

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

The sum of the first half is $a[0]+a[1]=5$
The sum of the second half is $a[2]+a[3]=7$
Therefore the stability is $5-7=-2$

Example 2: for the array

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

The sum of the first half is $a[0]+a[1]+a[2]=11$
Therefore the stability is $11-10=1$

**Part 1 (15 points)**

Write the following function:

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

Its parameters are an array of integers, `a`, and its length, `n`. the function returns the **stability** of the array.

**You are not allowed to use recursion at all.**
Part 2 (20 points)

Write the following recursive function:

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

Its parameters are an array of integers, `a`, and its length, `n`, the function returns the stability of the array.

**You are not allowed to use loops at all.**