Introduction To Computer – C (234112)

Spring 2011

Moed A

Duration: 150 minutes.
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 (20 points, 2 points for each output)

Complete the following output:

```c
#include <stdio.h>
int y = 10;

void f4(int *w) {
    *w = 1;
}

void f3(int *x) {
    *x = 2;
}

void f2(int x, int *z) {
    f3(&x);
    f4(&y);
    *z = 3;
    printf("x=%d y=%d z=%d\n", x, y, *z);
}

void f1(int x, int z) {
    x = 10;
    y = 11;
    z = 12;
    printf("x=%d y=%d z=%d\n", x, y, z);
}

int main(void) {
    int x = 9;
    int y = 9;
    printf("x=%d y=%d\n", x, y);
    /* output: x=_____ y=______ */

    f1(x,y);
    /* output: x=____ y=____ z=______ */

    printf("x=%d y=%d\n", x, y);
    /* output: x=_____ y=______ */

    f2(x,&y);
    /* output: x=____ y=____ z=______ */
    return 0;
}
```
**Question 2 (18 points, 3 points for each expression)**

Complete the missing expressions, so that the functions return an answer to the following question: "Is the array sorted in ascending order?" (ascending meaning $<$, not $\leq$), i.e. returning 1 if the array is sorted in ascending order, and 0 if not.

Note that the functions are separate and don't call each other.

**Examples:**
For the following array $a$:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>2</td>
<td>3</td>
<td>7</td>
<td>11</td>
<td>12</td>
<td>14</td>
</tr>
</tbody>
</table>

The function will return 1.

For the following array $a$:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>11</td>
<td>12</td>
<td>14</td>
</tr>
</tbody>
</table>

The function will return 0, because it is not ascending (3 appears twice in a row)

```c
int ok1(int a[], int n){
    int i;
    for (i = 1; i < n; ++i)
        if (_____________________
            ______________________;
            _____________;
}
```

```c
int ok2(int a[], int n){
    /* Fill the expressions, using recursion */
    if (n <= 1) return 1;
    else return (___________) && (___________);
}
```

```c
int ok3(int a[], int n){
    while ((--n > 0) && _____________);
    return n <= 0;
}
```
Question 3 (17 points, 5 for the first and 12 for the second)
Complete the following functions in a way that they return an answer that matches the comment in the beginning of the function. You are not allowed to declare variables in addition to the ones declared.

```c
int count_evens(int a[], int n){
    /* (5 points) the function returns the amount of even numbers in the array*/
    int i, count;
    return count;
}

int max_even_raw(int m[100][500]){  
    int i, max, i_max, count;
    /* (12 points) this function uses the previous function and returns an index of the row in the matrix which contains the most even numbers. If there are several lines with the most even numbers, return the index of the first one amongst them. */
return i_max;
}
```
**Question 4 (20 points)**

Write the following function:

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

Its parameters are an array of integers, \( a \), and its size, \( n \). The function needs to remove the zero’s from the array and "shrink" it, and to return the new size of the array. **The order of the elements different from 0 should be the same.**

**Example:** For \( n=9 \) and the array

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>3</td>
<td>-2</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

The function will return 5 (as there are five elements), and alter \( a \) to be:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>-2</td>
<td>7</td>
<td>2</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

The values in the cells not used in the array are insignificant. (cells 5 to 8 in the example)

```c
int remove_zeros(int a[], int n){
}
```
Question 5 (25 points)

In this question you need to use the **partition method (binary search)** in every part.

A `zeros_ones` array contains `n` numbers, and every number is either 0 or 1. The array starts with a series of 0's, and continues with a series of 1's. There is guaranteed to be at least one 0 in the array.

Complete the functions in the following parts, so that they return the number of zeros in the array.
You are not allowed to declare variables in addition to the ones declared.

**15 points – complete without using recursion**

```c
int number_of_zeros( int zeros_ones[], int n){
    int bottom = 0, top = n-1, mid;
    // Code here
}
```

**10 points – complete by using recursion**

```c
int number_of_zeros( int zeros_ones[], int n){
    int mid = n/2;
    if (n <=1) return n;
    // Code here
}
```