Carleton University
Department of Systems and Computer Engineering
SYSC 2006 - Foundations of Imperative Programming - Fall 2014

Section B - Midterm Exam

Instructions

1. This exam is closed book. Calculators are permitted.
2. Exam questions will not be explained and no hints will be given. If you think something is unclear or ambiguous, make a reasonable assumption (one that does not contradict the question), write it at the start of your solution, and answer the question. Please, do not ask the TAs questions during the exam.
3. Answer the questions in your answer booklet. You can write your solutions in pen or pencil.
4. You do not need to copy any code from the question paper to your answer booklet. You do not have to write comments in your C code; however, feel free to add comments to clarify anything you think requires further explanation. Please don't write comments that state the obvious; for example, there is no need for comments similar to this one:

   ```
   x = x + 1; // Add 1 to x
   ```
5. You can take this question paper with you when you leave the exam room.

PART A: Programming

Question 1 [4 marks]

Here is the declaration of a C structure that represents the coordinates of two-dimensional points:

```
struct point {
    int x;
    int y;
};
typedef struct point point_t;
```

Recall that the distance of a line between a pair of points ($x1$, $y1$) and ($x2$, $y2$) is given by the formula:

$$distance = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

Write the definition of a function named `distance` that is passed two points and returns the distance of the line between the points. The function prototype is:

```
double distance(point_t start, point_t end);
```

C's math library (`math.h`) has two functions that may be of use:

- `double pow(double x, double y);` returns the value of `x` raised to the power of `y`.
- `double sqrt(double x);` returns the square root of `x`.

You do not have to write `#include` statements in your solution.

Question 2 [11 marks]

(a) (1 mark) Here is the C statement that outputs an integer variable `k`, followed by the newline character (`'\n'`), which moves the cursor to the start of the next line:

```
printf("%d\n", k);
```

Here is a C statement that initializes an array named `samples` with six integers:

```
int samples[] = {5, 3, 9, 1, 7, 12};
```

Write the C statement that outputs the third integer (9) from array `samples`, followed by newline. Don't write a complete function; instead, just write the one C statement that is required.

(b) (2 marks) Assume that an array named `elems` has been initialized with `n` integers.Write the C statements that output each integer in array `elems` on a separate line. Don't write a complete function. Don't write the statements that declare and initialize `elems`.

(c) (2 marks) An array named `values` contains `n` integers. Write the C statements that output the difference between successive pairs of integers in array `values`; that is, the difference between the first and second integers, followed by the difference between the second and third integers, and so on. For example, if `values` and `n` are initialized this way:

```
int values[] = {5, 3, 9, 1, 7, 12};

n = sizeof(values) / sizeof(int); // n will be assigned 6
```

your code should output:

```
2
-6
8
-6
-5
```

Nothing should be output if `values` contains fewer than two integers.

Don't write a complete function; instead, just write the C statements that are required.

(d) (6 marks) A sequence of integers is an arithmetic sequence if the difference between successive integers is always the same. A sequence with one or two integers is considered to be an arithmetic sequence. For example:

- The sequence `3` is an arithmetic sequence;
- The sequence `4, 7` is an arithmetic sequence;
- The sequence `4, 7, 10, 13, 16` is an arithmetic sequence (the difference between successive numbers is always `3`);
- The sequence `3, 6, 9, 11, 14` is not an arithmetic sequence.

Write a function named `arithmetic_sequence` that is passed an array containing a sequence of `n` integers. The function prototype is:

```
_Bool arithmetic_sequence(int seq[], int n);
```

The function returns `true` if the integers form an arithmetic sequence; otherwise it returns `false`. Assume that parameter `n` will always be positive. (Your function does not have to check this).

PART B: Tracing Code/Memory Diagrams

Question 3 [7 marks]

```
#include <stdlib.h>

int mystery(int k, int *p)
{
    int m;
    m = 5 * k - *p;   /* Point A */
    k = k + 3;
    return m;

}

int main(void)
{
    int j = 8;
    int k = 2;

    int result = mystery(j, &k);  /* Point B */
    exit(0);
}
```

Using the notation presented in lectures, draw two separate memory diagrams, one each for parts (a) and (b). *Do not combine your solutions into a single diagram.* Do not use arrows to depict the flow of data into and out of variables. Remember, arrows are only used to depict pointers.

(a) Draw a diagram that depicts the program's activation frame(s) immediately **after** the statement at Point A is executed; that is, immediately after `m = 5 * k - *p;` is executed, but before `k = k + 3;` is executed.

(b) Draw a diagram that depicts the program's activation frame(s) immediately **after** the statement at Point B is executed; that is, immediately after `int result = mystery(j, &k);` has been executed, but before `exit` is called.

Question 4 [8 marks]

```
#include <stdlib.h>

void strange(int arr[], int n)
{
    int *ptr = arr;
    int i;

    /* Point A */
    for (i = 0; i <= (n - 1) / 2; i = i + 1) {
        *(arr + i) = ptr[0] * 3;
        ptr = ptr + 2;
    }
    /* Point B */
    return;
}

int main(void)
{
    int a[] = {4, 2, 8, 5, 3};
    strange(a, 5);
    exit(0);
}
```

Using the notation presented in lectures, draw two separate memory diagrams, one each for parts (a) and (b). *Do not combine your solutions into a single diagram.* Do not use arrows to depict the flow of data into and out of variables. Remember, arrows are only used to depict pointers.

(a) Draw a diagram depicting this program's activation frame(s) when execution reaches Point A; that is, just **before** the `for` loop is executed.

(b) Draw a diagram depicting this program's activation frame(s) when execution reaches Point B; that is, just **after** execution of the `for` loop has finished, but before the `return` statement is executed.

When answering this part, make sure that you clearly separate your rough work from your final solution. Consider sketching a sequence of diagrams (one for each iteration of the loop) on the unlined (rough work) pages of your answer booklet, and then draw the final diagram (showing the state of the program at Point B) on a lined page.