UNIVERSITY OF TORONTO

Faculty of Applied Science and Engineering

APS 105H1S

Final Examination

Instructor: J. Deber

April 29, 2011 9:30 a.m. - noon

Duration — 2.5 hours

Exam Type: A (no external aids) / Calculator Type: 4 (no electronic devices)

Student Number:	
Last (Family) Name(s):	
First (Given) Name(s):	

---

*Do **not** turn this page until you have received the signal to start.*
(In the meantime, please fill out the identification section above, and read the instructions below *carefully*.)

This final examination consists of 10 questions on 20 pages (including this one). *When you receive the signal to start, please make sure that your copy of the examination is complete.*

You must write your programs in the C programming language, and you may make use of any C features or libraries that we have covered. Standard assignment style rules (including a ban on `break`, `continue`, and certain string functions) apply, unless indicated otherwise. You may make use of C99 features.

You do not have to provide any `#include` statements.

Comments are not required, although we will be marking style on coding questions.

Unless indicated otherwise, you do not have to handle abnormal input (e.g., a user entering a letter when a number is expected).

Unless indicated otherwise, you may write helper functions. Helper functions can be reused in any part of a multi-part question.

If you use any space for rough work, indicate clearly what you want marked.

There is a function reference sheet on page 17.

MARKING GUIDE

# 1:	______/ 5
# 2:	______/ 8
# 3:	______/ 6
# 4:	______/ 7
# 5:	______/11
# 6:	______/ 8
# 7:	______/ 7
# 8:	______/10
# 9:	______/19
# 10:	______/ 5
TOTAL:	______/86

Good Luck!

Question 0.

Write your student number in the space provided at the bottom of each odd-numbered page. Failure to do so will result in a 2 mark deduction.

Question 1. [5 MARKS]

Consider the following program:

```
void g(int i)
{
    printf("%d\n", i);
}

void f(int i, int j)
{
    printf("%d, %d\n", i, i+j);
    g(j);
}

int reset(int i)
{
    i = 0;
    printf("%d\n", i);
    return i;
}

int main (void)
{
    char c = 'J';
    double d = 1.8;
    int i = 0;

    for (int j = 0; j < d; i++, j++)
    {
        printf("%c\n", c);
        f(i, j);
        reset(j);
    }

    printf("%d\n", i);

    return 0;
}
```

Write the output of this program. You may use the space provided below or to the right of the program.

Question 2. [8 MARKS]

Part (a) [4 MARKS]

Write the *pseudocode* for bubble sort:

Part (b) [3 MARKS]

Given the following set of ints, illustrate the progression of a bubble sort in ascending order. In other words, every time that a number is moved, write down the new configuration of the list. In addition, you should clearly indicate which configurations belong to which iteration of the bubble sort by drawing a horizontal line between passes.

`16, 128, 2, 8, 4` (initial order)

`2, 4, 8, 16, 128` (final sorted order)

Part (c) [1 MARK]

How many times does bubble sort need to traverse this list?

Question 3. [6 MARKS]

The following code has several mistakes in it. These mistakes include syntax errors, logic problems (i.e., bugs), and style issues.

One mistake (misspelling int) is already circled.

```
#define ARRAY_SIZE = 3;

char *func(*s, n);

1 in main(void)
{
    int a[] = malloc(ARRAY_SIZE);
    for (i = 0; i <= ARRAY_SIEZ; i++)
    {
        a[i] = rand() % 5;
    }

    printf("The list of numbers is: %d\n', a);

    for (i = 0; i < ARRAY_SIZE; i++)
    }
        char *rep = func("*", a[i])
        printf(rep);
    }
}

void func(char *s, int n)
{
    char *result = malloc(n * sizeof(char *));
    for (int i = 0; i < n; n++)
    {
        strcat(result, s);
    }

    return s;

    s[strlen(s)] = \0;
}
```

Part (a) [1 MARK]

What is this code trying to do?

Part (b) [5 MARKS]

Identify the mistakes in the code on the previous page. For each mistake, you should circle the relevant section of code, and write a number next to the circle. Then, in the space provided below, you should briefly describe each mistake and provide a possible correction.

A mistake (misspelling `int`) is already circled, numbered, and described to demonstrate the format you should use.

Note: there are many mistakes in this code; to earn full marks, you will need to correctly identify (and provide the appropriate correction for) 10 mistakes in addition to the one provided below.

1. Return type of `main()` is misspelled `in` rather than `int`.

Question 4. [7 MARKS]

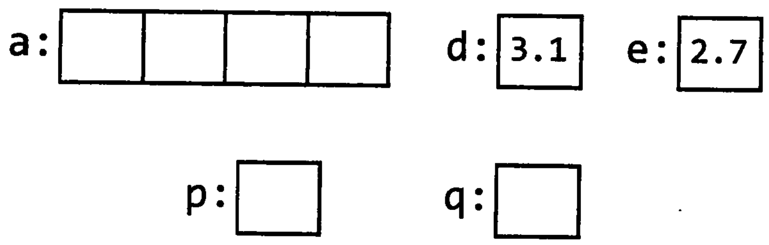
Consider the following code:

```
double d = 3.1;
double e = 2.7;
double a[4];
double *p;
double *q;

/* Memory snapshot taken here. */

p = &d;
a[1] = *p;
q = a;
*q = e;
*(q + 2) = (d + 2);
*p = *q;
p = q + (int)(*p);
```

A snapshot of memory after the first five lines have executed is pictured below:



For each of the statements listed on the next page, fill in the values that each variable would have *after* that statement has executed. Pointers should be drawn as arrows.

Note that the statements are cumulative; changes to variables that are made on one line persist to the next statement.

`p = &d;`	a: [ ][ ][ ][ ] d: [ ] e: [ ] p: [ ] q: [ ]
`a[1] = *p;`	a: [ ][ ][ ][ ] d: [ ] e: [ ] p: [ ] q: [ ]
`q = a;`	a: [ ][ ][ ][ ] d: [ ] e: [ ] p: [ ] q: [ ]
`*q = e;`	a: [ ][ ][ ][ ] d: [ ] e: [ ] p: [ ] q: [ ]
`*(q + 2) = (d + 2);`	a: [ ][ ][ ][ ] d: [ ] e: [ ] p: [ ] q: [ ]
`*p = *q;`	a: [ ][ ][ ][ ] d: [ ] e: [ ] p: [ ] q: [ ]
`p = q + (int)(*p);`	a: [ ][ ][ ][ ] d: [ ] e: [ ] p: [ ] q: [ ]

Question 5. [11 MARKS]

Assume that you have sorted linked lists built out of the following `struct`:

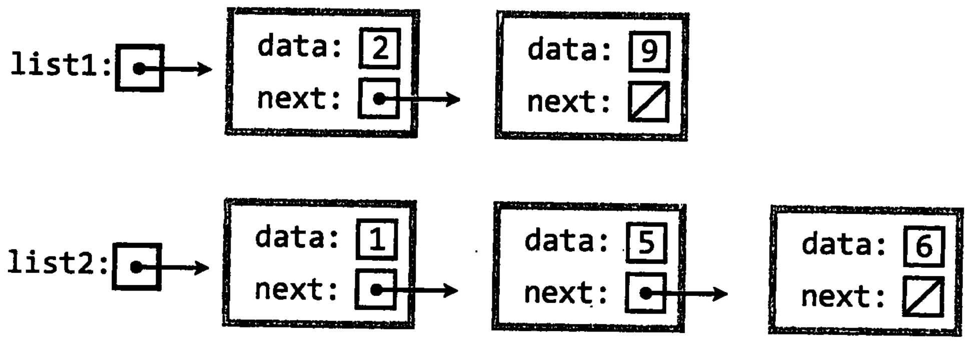
```
typedef struct node
{
    int data;
    struct node *next;
} Node;
```

A sorted linked list is a linked list in which the nodes are arranged so that each node in the list has a `data` that is smaller than its predecessor (i.e., `current.data < current.next.data` is true for all nodes except the last node (since its `next` is `NULL`)). You may assume that there are no duplicate entries (i.e., there is at most one node with any given value for `data`).

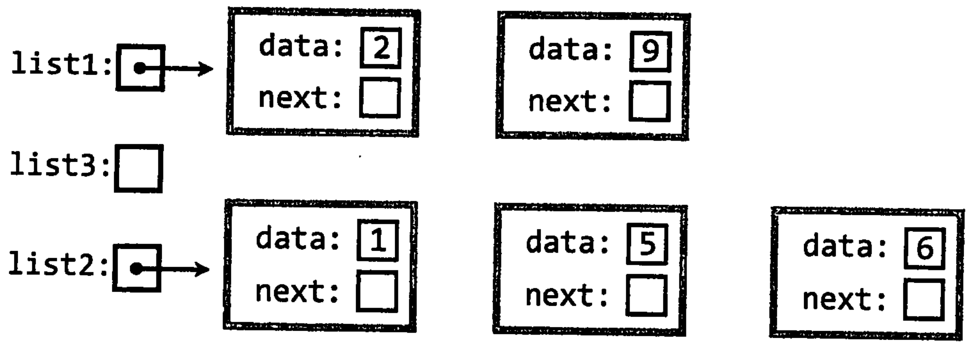
Merging two lists means interleaving the two sorted linked lists so that they form a single sorted linked list. This process destroys the two original lists, since the `next` pointers are overwritten when creating the new list.

Part (a) [2 MARKS]

Consider the following two linked lists:



Draw the linked list `list3` that would result from merging them:



Part (b) [7 MARKS]

Write the function documentation, prototype, and *pseudocode* for a function that takes two sorted linked lists, and merges them together into a single list. Make sure you clearly indicate what (if anything) the function returns, and what (if anything) it takes as a parameter(s). You can assume that each list has at least one `Node` in it. *Note: you only need to write the pseudocode for this function; you do **not** need to write an implementation of it in C.*

Part (c) [2 MARKS]

Given two linked lists `list1` and `list2`, write a code fragment that uses your function from Part (b) to create a new linked list called `list3` that is the result of merging `list1` and `list2`.

Question 6. [8 MARKS]

Consider the following function:

```
/* Starting at the first element, writes every other element of 'a' (which
 * is 'n' elements long) into 'new' (which is X elements long). */
void everyOtherElement(const int a[], int n, int new[]);
```

For example, the array {2, 5, 8, 9, 10} would produce {2, 8, 10}, and the array {45, 20, 18, 95} would produce {45, 18}.

Part (a) [1 MARK]

In the above documentation, X (the length of new) is not defined. Write an expression that calculates X in terms of n.

Part (b) [5 MARKS]

Write the function:

```
void everyOtherElement(const int a[], int n, int new[])
{
```

Part (c) [2 MARKS]

Provide a code fragment that uses everyOtherElement() to generate the array that consists of every other element of array1, prints out the resulting array, and then performs any necessary cleanup.

```
int array1[] = {8, 32, 43, 88};
```

Question 7. [7 MARKS]

Part (a) [5 MARKS]

Write the following function:

```
/* Creates and returns a new string made up of the first 'n' characters of 's1' followed by
 * the first 'm' characters of 's2'.  's1' and 's2' must be at least 'n' and 'm' characters
 * long, respectively (excluding the null characters).  The resulting string must subsequently
 * be destroyed by a call to free(). */
char *concatPortions(const char *s1, int n, const char *s2, int m)
{
```

Part (b) [2 MARKS]

Provide a code fragment that uses `concatPortions()` to create a string consisting of the first 2 characters of `"exact"` and the first 2 characters of `"ambient"`, prints it out, and performs any necessary cleanup.

Question 8. [10 MARKS]

In this question you will write two different functions that reverse a string. A reversed string is the string obtained by starting at the rightmost character, and reading right-to-left. For example, the reverse of `"abc"` is `"cba"`, and the reverse of `"hello123"` is `"321olleh"`.

Part (a) [5 MARKS]

Write `reverse1()`, a `void` function that takes a single string parameter, and overwrites that string with its reversed version. You must write the function iteratively (i.e., using a loop); any solution that uses recursion will receive a mark of 0.

Part (b) [5 MARKS]

Write `reverse2()`, a `void` function that does the same thing as `reverse1()`. `reverse2()` must be written recursively; any solution that uses a loop will receive a mark of 0.
Hint 1: you may want to write a recursive helper function.
Hint 2: this is similar to some examples we have examined in class.

Question 9. [19 MARKS]

Part (a) [2 MARKS]

Write a definition of a `struct` that can store information about a book. The `struct` should be able to store the book's title, the author's last name, and the book's weight (in kg). You are free to use whatever data types you think are appropriate for each variable. The `struct` should be `typedef`'ed with the name `Book`.

Part (b) [3 MARKS]

Write a function called `createBook()` that takes three parameters (the title, last name, and weight of the book), creates a new `Book`, sets its fields to the specified values, and returns a pointer to it.

Part (c) [2 MARKS]

Write a `void` function called `destroyBook()` that takes a pointer to a `Book` as a parameter, and performs any tasks necessary to cleanup that `struct`.

Part (d) [5 MARKS]

Write the following function:

```
/* Compares 'b1' and 'b2', and returns -1 if 'b1' < 'b2', 0 if 'b1' == 'b2',
 * and 1 if 'b1' > 'b2'.  Compares Books based on the lexicographical ordering
 * (i.e., ASCIIbetical dictionary ordering) of the authors' last names.  If the
 * authors' last names are the same, we then compare the titles.  If both the authors'
 * last names and the titles are identical, we consider the two books to be identical. */
int compareBooks(const Book *b1, const Book *b2);
```

For example, a book by Carter would be ordered before a book by King, independent of the title. A book by Smith titled *A C Book* would be ordered before another book by Smith titled *The Best C Book*.

Part (e) [3 MARKS]

Write a code fragment that creates two Books: *The C Programming Language* by Kernighan and Ritchie (you can treat the string "Kernighan and Ritchie" as a single name) that weighs 0.43 kg, and *An Introduction to Computer Science* by Carter that weighs 0.61 kg. Your code should then compare the two Books, and print the title of the one that would be ordered second. Finally, you should cleanup the two Books.

Part (f) [4 MARKS]

Write the following function:

```
/* Computes and returns the total weight of all of the Books in 'cart', which is an 'n' element
 * array of pointers to Books. */
double calculateWeight(Book *cart[], int n)
{
```

Question 10. [5 MARKS]

Based on your experience programming in C this semester, if you were given the opportunity to change one thing about the C programming language, what would it be? This could include adding or removing features, changing how the language behaves, modifying the syntax, changing the standard library, etc.

Justify your answer, and be specific – a response of "it should be easier" will not earn many marks.

Function Reference Sheet

(Note: you are welcome to use functions that are not listed on this page, unless they have been expressly prohibited)

```
/* Copies at most 'n' characters from 'src' (including the terminating '\0' character) to 'dest'. */
strncpy(char *dest, const char *src, size_t n);

/* Appends at most 'n' characters from 'src' to 'dest', and then adds a terminating '\0' character. */
strncat(char *dest, const char *src, size_t n);

/* Calculates the length of 's', not including the terminating '\0' character. */
size_t strlen(const char *s);

/* Compares the two strings 's1' and 's2'.  Returns an integer less than, equal to, or greater than zero if 's1' is found
 * to be less than, equal to, or greater than 's2'. */
int strcmp(const char *s1, const char *s2);


/* Allocates 'size' bytes, and returns a pointer to the allocated memory. */
void *malloc(size_t size);

/* Frees the memory space pointed to by 'ptr'.
void free(void *ptr);

/* Changes the size of the memory block pointed to by 'ptr' to 'size' bytes, and returns a pointer to the allocated memory. */
void *realloc(void *ptr, size_t size);


/* Reads in at most 'size' - 1 characters from 'stream', and stores them in 's'.  Reading stops after a newline, which is
 * stored in 's'.  's' will be null terminated. */
fgets(char *s, int size, FILE *stream);


/* Returns a pseudo-random int between 0 and RAND_MAX. */
int rand(void);

/* Sets the seed of the PRNG to 'seed'. */
void srand(int seed);

/* Returns the cosine of 'x', which is in radians. */
double cos(double x);

/* Returns the sine of 'x', which is in radians. */
double sin(double x);

/* Returns the tangent of 'x', which is in radians. */
double tan(double x);

/* Rounds 'x' to an integer value. */
double rint(double x);

/* Rounds 'x' up to the nearest integer value. */
double ceil(double x);

/* Rounds 'x' down to the nearest integer value. */
double floor(double x);

/* Returns the absolute value of 'x'. */
int abs(int x);

/* Returns the absolute value of 'x'. */
double fabs(double x);

/* Returns the natural logarithm of 'x'. */
double log(double x);

/* Returns e raised to the power of 'x'. */
double exp(double x);

/* Returns 'x' raised to the power of 'y'. */
double pow(double x, double y);

/* Returns the square root of 'x'. */
double sqrt(double x);


/* Checks if 'c' is an alphabetic character. */
int isalpha(int c);

/* Checks if 'c' is a digit. */
int isdigit(int c);

/* Checks if 'c' is an upper case letter. */
int isupper(int c);

/* Checks if 'c' is a lower case letter. */
int islower(int c);

/* Converts 'c' to upper case, if possible. */
int toupper(int c);

/* Converts 'c' to lower case, if possible. */
int tolower(int c);
```

[You may use these pages for either rough work or as additional space for your solutions to other questions. These pages will not be marked, unless you clearly indicate the part of your work that you want us to mark.]

[You may use these pages for either rough work or as additional space for your solutions to other questions. These pages will not be marked, unless you clearly indicate the part of your work that you want us to mark.]