UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING

APS 105 — Computer Fundamentals
Final Examination
April 21, 2017
2:00 p.m. – 4:30 p.m.
(150 minutes)
Examiners: B. Li, J. Rose and M. Stumm

Exam Type A: This is a "closed book" examination; no aids are permitted.

Calculator Type 4: No calculators or other electronic devices are allowed.

All questions are to be answered on the examination paper. If the space provided for a question is insufficient, you may use the last page to complete your answer. If you use the last page, please direct the marker to that page and indicate clearly on that page which question(s) you are answering there.

You must use the C programming language to answer programming questions. You are not required write `#include` directives in your solutions. Except those excluded by specific questions, you may use functions from the `math` library as necessary.

The examination has 22 pages, including this one.

First Name: _____ Last Name: _____

Student Number: _____

MARKS

1	2	3	4	5	6	7	8	9	10	11	12	13	14	Total
/2	/2	/2	/2	/2	/3	/2	/8	/7	/8	/10	/10	/10	/10	/78

Question 1 [2 Marks]

Find and correct all compile-time errors (mistakes that would cause compilation or that would cause the 'build' to fail) in the following C program. Your answer should both identify what the error is, and what the correction should be. Marks will be deducted if you identify correct items as compile-time errors.

```
#include <stdio.h>
int main(void) {
  int j, k;
  int *i = &j;

  for (*i = 0; *i < 10, *i = *i + 1) {
    scanf("%d", &k);
    printf("%d", (*i) * (*i) * (*i));
  }
}
```

Answer:

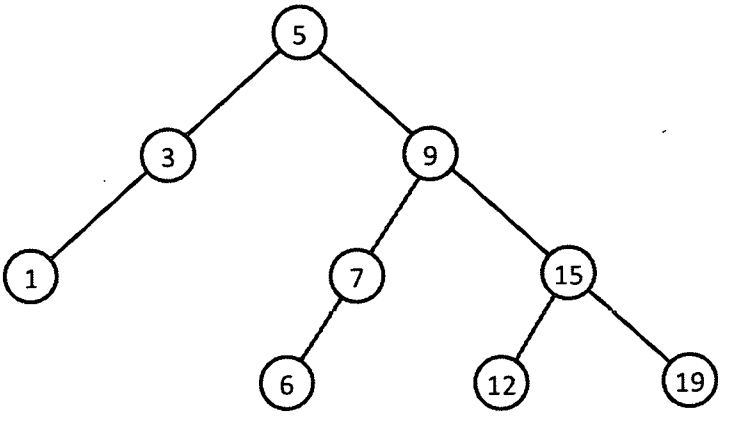
Question 2 [2 Marks]

Write a single C statement that declares an `int` variable `randomChoice`, and initializes it with a number that is randomly selected from the following set: `-5, 5, 0, -10, 10`.

Answer:

Question 3 [2 Marks]

The following binary search tree was constructed by inserting a sequence of values into an empty tree:



Which of the following insert sequences will **not** produce the above binary search tree:

A: 5 3 9 1 7 15 6 12 19

B: 5 3 1 9 15 19 7 6 12

C: 5 9 7 6 15 12 3 19 1

D: 5 9 6 7 15 12 19 3 1

E: 5 9 3 7 1 15 19 12 6

(Please note the "not" above.)

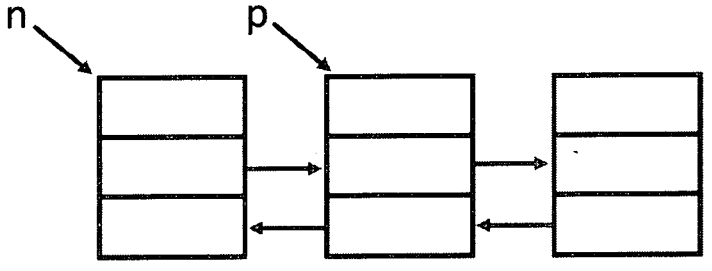
Answer:

Question 4 [2 Marks]

In a *doubly* linked list, each node has two pointers: one called `next` which points to the next node in the list, and one called `prev` which points to the previous node in the list, as in the following declaration:

```
struct node {
  int key;
  struct node *next;
  struct node *prev;
} *n, *p;
```

Consider a part of a doubly linked list as shown below where the two variables, `n` and `p`, have been set to point to nodes as shown in the figure:



Which of the following expression(s) does **not** refer (point) to the third, right-most node in the diagram above?

A: `p->next`

B: `n->next->next`

C: `p->next->prev`

D: `n->next->prev->next->next`

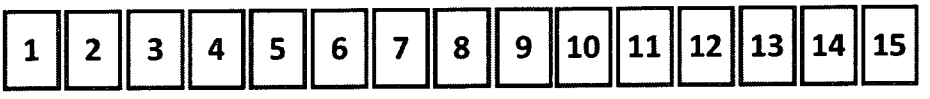
E: `p->next->prev->next`

(Again, please note the "not" above.)

Answer:

Question 5 [2 Marks]

Consider the following array with exactly 15 elements:



Suppose we are doing a binary search on the array.

Circle all elements that would **not** be found after *examining* three numbers if we were searching for the value of the target element using binary search. (An element is considered "examined" if it is compared against the value being searched for.)

Answer:

Question 6 [3 Marks]

Recall the *cocktail* sort method that you used in Lab 9 in this course, and consider the following sequence of numbers:

```
8  6  5  3  9  12
```

Give the result of the first three passes of the cocktail sort algorithm on these numbers, assuming that the sorted order is to have the lowest number on the left, and that the sorting algorithm starts on the left end of the above numbers. A *pass* is one traversal through the set of numbers.

Answer:

After Pass 1: _____

After Pass 2: _____

After Pass 3: _____

Question 7 [2 Marks]

There have been four plenary lectures since the midterm in this course, and there is one question from each of these lectures below. You must answer **two of these four questions**. Each answer should be one or two sentences. If you answer more than two questions, you must indicate which two you wish to be graded; if you do not indicate which, then the first two in order will be graded.

Plenary Lecture 6 — From Circuits to Software (Professor Jason Anderson)
Professor Anderson described methods of converting computations (that are described in software) into hardware that could perform that same computation. For example, an addition in software (such as `a = b + c;`) would be converted into a hardware *adder*. Similarly multiplication and division are converted into multipliers and dividers. He also described a key step in the overall process of conversion, which was called *scheduling*. What is the purpose of the scheduling step?

Plenary Lecture 7 — Machine Learning and Artificial Intelligence (Xavier Snelgrove)
Mr. Snelgrove described two methods for having a computer try to understand human language. The first method used 'if' statements, like those used in this course, to identify specific words and phrases that are present in the words written. What is the problem with this method, and what better method did he describe for performing this function?

Plenary Lecture 8 — Computer Vision (Professor Sven Dickinson)
Professor Dickinson described many possible applications that use *computer vision* methods. Name two of them.

Plenary Lecture 9 — Accenture (Safdar Mahmood)
Mr. Mahmood described the notion of 'disruption' in the modern, business context. What does the word disruption mean in that context?

Question 8 [8 Marks]

Write a C language function called `rotateAndFindLongest` that does two things:

(a) It *rotates* the values of four integer variables that exist in the calling function. For example, if the value of the four variables in the calling function were 34, 28, 66, 19 before the function was called, the variable values would be 19, 34, 28, 66 after the function was called.

(b) The function should return the value of the largest of the all four variables.

Answer:

Question 9 [7 Marks]

This question has several, somewhat related, parts:

(a) Write the C-language code to declare four variables with the following types. You can freely choose the name of your variables.

- A 32 bit signed integer
- A double
- A char
- An array of bool of size 10

Answer:

(b) Write the complete C-language code that declares a struct, called **myStruct**, that contains all four of the types from part (a) as members.

Answer:

(c) Write the C-language code for a new version of the struct in part (b), called **myLLStruct**, that would enable it to be used as a node in a linked list.

Answer:

(d) Write a single C statement to declare an array (called **myArrayStruct**) of size 100 of the struct declared in **part (b)** - i.e of the struct **myStruct**.

Answer:

(e) Write several lines of C-language code that will initialize the first 50 integers in the above array (from part (d)) to the value 1000, and the last 50 doubles to be 0.5, and every boolean value in the array to be **false**. All other values in the array should *not* be initialized.

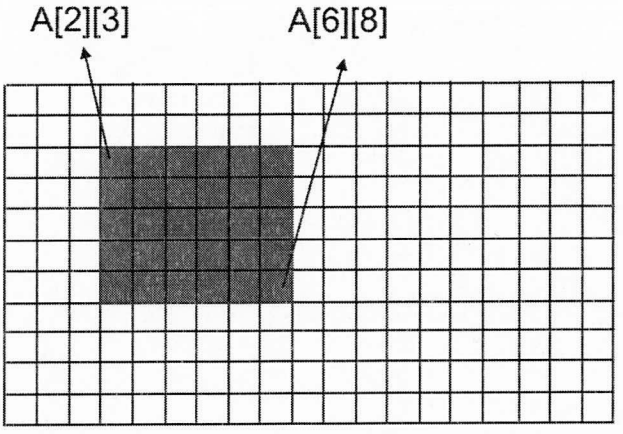
Answer:

Question 10 [8 Marks]

You are to write a function that sets a rectangular portion of a 2-dimensional array to a specific value. The function takes the following as input parameters: an integer 1000x1000 2-dimensional array called `A`, four integer parameters named `rowStart`, `rowEnd`, `colStart`, and `colEnd` and an integer parameter `value`.

The purpose of the function is to set the value of the array elements in `A` beginning with `A[rowStart][colStart]` and ending with `A[rowEnd][colEnd]` to be the value `value`. The picture below illustrates that the shaded portion of the array should have its elements changed to `value`, for the specific example given:

rowStart = 2, rowEnd = 6, colStart = 3, colEnd = 8



However, this function is to be used by students in a first year C programming course, and those students are well-known for writing code that has errors. For example, they often attempt to access elements of an array outside of its defined bounds, among other problems. Therefore, your function should check that the inputs to the function are correct before performing the above operation. You must determine, through the reading of this question, what would make the inputs `rowStart`, `rowEnd`, `colStart`, `colEnd` and `value` correct. (You can assume that the array `A` is passed in correctly.)

The function should return a boolean result that is set to `true` if the inputs are valid, and `false` if they are not. In the case that the inputs are not valid, the setting of the values of the array should not be attempted.

IMPORTANT: Your solution, in addition to the code for the function described above, should also show how you would call this function from the main function, **and make use of its return value**. That is, you should show the call to the function (with appropriately declared variables, and gathering input from the user for `rowStart`, `rowEnd`, `colStart`, `colEnd` and `value`) and make appropriate use of the returned boolean value.

(give your answer on next page)

Answer: *give your solution from the above question on this page.*

Question 11 [10 Marks]

Write a function `char *deleteSubString(char *source, char *substring)` that takes two strings called `source` and `substring` as its parameters. It should return a new *dynamically allocated* string that is constructed from the strings `source` and `substring`. The newly created and returned string should be the same as the `source` string, but with the **first** occurrence of the string `substring` removed. For example, if the string `source` has the value `"my toronto"`, and the string `substring` has the value `"to"`, the function will return the string `"my ronto"`. You may use any of the string-related functions in the C standard library, and may assume that `string.h` has been included.

Answer: *(You can continue your solution on the next page.)*

```
char *deleteSubString(char *source, char *substring) {
```

continue your solution from the above question on this page.

Question 12 [10 Marks]

Recall that, in a binary tree, a node that has no children is called a *leaf*. Given the following node declaration:

```
struct treeNode {
  int value;
  struct treeNode *left;
  struct treeNode *right;
};
```

write a function called `treeLeafCount()` that takes one `struct treeNode *root` parameter and returns the number of leaves in the tree pointed to by `root`. You may not use global variables in your solution.

Answer:

Question 13 [10 Marks]

QuickSort is considered one of the fastest sorting algorithms in practice. However, it turns out that Insertion Sort is faster than QuickSort for smaller arrays; e.g., for arrays with 10 or fewer elements. Because of this, many implementations use a combination of both algorithms: they use QuickSort when the size of the array segment to be sorted is larger than 10, but switch over to Insertion Sort when the size of the array segment to be sorted is less than or equal to 10.

Your job is to implement QuickSort for an array of doubles that automatically switches over to Insertion Sort for the small array segments with less than or equal to 10 elements. To make your job easier, you can assume the following function is available:

```
int selectPivotAndPartition(double list[], int from, int to);
```

This function processes the segment of array `a` from index `from` to index `to`. It selects a pivot in a smart way, and then partitions all elements in the `[from, to]` segment so that all elements less than or equal to pivot are located to the left of the pivot element and all elements greater than or equal to pivot are located to the right of the pivot element. The function then returns the index of the array where the pivot is located. Hence, after you call

```
int pIndex = selectPivotAndPartition(list,  from, to);
```

you are guaranteed that

`list[i] <= list[pIndex]` when $\texttt{from} \leq \texttt{i} \leq \texttt{pIndex}$

and

`list[i] >= list[pIndex]` when $\texttt{pIndex} \leq \texttt{i} \leq \texttt{to}$

Give your answer on the next pages.

Answer:

```
void quickSort(double list[], int from, int to) {
```

Continue your solution to the previous question on this page.

Question 14 [10 Marks]

The following C structure is used to define each node in a linked list:

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

Write a C function, called `buildJoinedList`, that takes two linked lists called `firstList` and `secondList` as its parameters, and returns a new list that joins the two lists, with `secondList` at the front. Both `firstList` and `secondList` are pointers to the first node of a linked list. The function should return a pointer to a new list that is dynamically allocated.

Note that the existing linked lists pointed to by `firstList` and `secondList` must not be modified in any way.

An example of how the function should work is as follows: if `firstList` points to a linked list containing nodes storing the numbers `1, 2, 3, 4, 5` and `secondList` containing the numbers `6, 7, 8, 9, 10`, then the newly created list returned by the `buildJoinedList` function should contain nodes storing the numbers `6, 7, 8, 9, 10, 1, 2, 3, 4, 5`.

Answer: *(You can continue your solution on the next page.)*

Continue your solution from the above question on this page..

This page has been left blank intentionally. You may use it for answers to any question in this examination.