UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING

FINAL EXAMINATION, APRIL 2023

DURATION: 2½ hours

ESC 190 H1S — COMPUTER ALGORITHMS AND DATA STRUCTURES

Calculator Type: None
Exam Type: D

Aids allowed: reference sheet distributed with the exam
Examiner(s): M. Guerzhoy

Student Number: |__|__|__|__|__|__|__|__|__|__|__|__|

UTORid: _____

UofT email: _____@mail.utoronto.ca

Family Name(s): _____

Given Name(s): _____

---

*Do **not** turn this page until you have received the signal to start.*
In the meantime, please read the instructions below *carefully*.

This final examination paper consists of 10 questions on 26 pages (including this one), printed on both sides of the paper. *When you receive the signal to start, please make sure that your copy is complete, and fill in the identification section above.*

Answer each question directly on this paper, in the space provided. Use the pages at the end of the exam for extra space. If you use extra pages, indicate that you have done so in the space under the question.

Write up your solutions carefully! Comments and docstrings are *not* required to receive full marks, except where explicitly indicated otherwise. However, they may help us mark your answers, and part marks *might* be given for partial solutions with comments clearly indicating what the missing parts should accomplish.

When you are asked to write code, *no* error checking is required: you may assume that all user input and argument values are valid, except where explicitly indicated otherwise.

When writing in Python 3 programming language. You may not `import` any module except `math`, unless otherwise specified. When writing in C, You may `#include` any header file that is available with a standard installation of `gcc` or is described in the C99 standard, **unless otherwise specified**. We will accept code that is correct under the C99 standard as well as any code that will run correctly when compiled with `gcc`. Write your student number on every odd-numbered page you use. Failure to do so will result in a penalty of 5 points.

MARKING GUIDE

# 1:	_____/ 10
# 2:	_____/ 10
# 3:	_____/ 10
# 4:	_____/ 10
# 5:	_____/ 15
# 6:	_____/ 15
# 7:	_____/ 10
# 8:	_____/ 10
# 9:	_____/ 10
# 10:	_____/ 10
TOTAL:	_____/110

Good Luck!

OVER...

Question 1. [10 MARKS]

Part (a) [8 MARKS]

In C, write a function that returns $n! = 1 \times 2 \times 3... \times n$. You may use either loops or recursion.

```
double fact(int n)
{
```

Part (b) [2 MARKS]

What is the runtime complexity of the function you wrote?

CONT'D...

Question 2. [10 MARKS]

In C, write a function which takes in a string, and replaces every occurrence of `"winter"` (in lowercase) with `"summer"` (in lowercase).

```
char s[] = "In the winter, I will rest and enjoy the sun. Winter is great!
Wait, actually I mean winter."
replace_ws(s);
// s is now "In the summer, I will rest and enjoy the sun. Winter is great!
// Wait, actually I mean summer."
// "Winter" with the upper-case W was not replaced.
// For this question, do not worry about newlines.


void replace_ws(char *s)
{
```

Question 3. [10 MARKS]

In C, write a recursive function that compares two blocks of integers and returns 1 if they are equal and 0 if they are not equal. For example, your function should be able to handle the following:

```
int arr1[] = {7, 8, 9};
int arr2[] = {7, 8, 9};
int arr3[] = {7, 8};
// Want to be able to use the recursive function to get 1 when comparing arr1 and arr2
// and to get 0 when comparing arr1 and arr3.
```

The name of the function as well as the function signature (i.e., what parameters the function takes) are up to you. You **may not** use any helper functions from inside the function you write. You **may not** use global variables. You **may not** use loops. Write a `main` function and show how you would use the function you wrote.

Question 4. [10 MARKS]

Write a function that inserts a node with the value `new_val` at index i of a linked list. You must use the definitions below. Make sure that the code works on lists of **any size**. If you need helper functions, you must write them yourself.

You may need to make assumptions like "the next field of the last node is NULL." State any such assumption as a comment.

For example, if you are inserting the value `42` at index `1` into the linked list represented by `10->20->30`, the linked list will become `10->42->20->30`.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

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


typedef struct LL{
  node *head;
  int size;
} LL;



// append integer new_elem to linked list my_list
void LL_append(LL *my_list, int new_val, int i)
{
```

CONT'D...

Question 5. [15 MARKS]

Consider the 2D Integer Matrix Abstract Data Type. The 2D Matrix is a rectangular array of integers of size $n \times m$. The operations are:

- getElem(M, i, j): get element at location (i, j) of matrix M
- Set element at location (i, j) of matrix M
- Return a new matrix whose value is the sum of the matrices A and B

Implement this ADT using the C programming language. Show how to use your implementation to compute

$$\begin{pmatrix} 1 & -1 \\ 2 & 0 \end{pmatrix} + \begin{pmatrix} -1 & 2 \\ 0 & 1 \end{pmatrix}.$$

Your implementation **must** allow the user to operate on matrices of any size. As long as you follow the specifications outlined in the question, you can make any choices you want and make any reasonable assumptions. Indicate which part of your code would go in which of the files `main.c`, `matrix.c`, `matrix.h`.

CONT'D...

Question 6. [15 MARKS]

Implement the Priority Queue ADT. You must implement the operations

```
insert(x, p): insert the element x whose priority is p
extract_min(): remove and return the element whose priority is smallest
```

There are no requirements regarding the efficiency of your code. We recommend implementing a simple but slow algorithm. You only need to implement the functions `insert` and `extract_min`. The design is up to you. You may use C or Python, but we recommend Python.

State what the complexity of your code is, and briefly explain your reasoning.

Question 7. [10 MARKS]

We would like to be able to sort lists of integers by their average. For example, if I have the following lists

```
1, 3 # average:2
6, 4, 5 # average: 5
20, -20 # average: 0
```

I would like to be able sort the lists as follows:

```
20, -20 # average: 0
1, 3 # average:2
6, 4, 5 # average: 5
```

We would like to be able to use the following:

```
my_lists = [MyList([1, 3]), MyList([5, 4, 6]), MyList([20, -20])]
print(sorted(my_lists)) # prints out [[-20, 20], [1, 3], [6, 4, 5]]
```

Write Python 3 code so that the two lines above run as specified in the comment, for any input lists of integers. Assume all lists are non-empty.

CONT'D...

Question 8. [10 MARKS]

Write a function in C code that takes in a string of decimal digits representing a number and returns a string of binary digits representing the same number.

For example,

```
dec2bin("11") // returns the string "1011"
```

```
char *dec2bin(const char *decimal)
{
```

Question 9. [10 MARKS]

You are given a staircase with n steps, and you can climb either 1, 2, or 3 steps at a time. Each step has an associated cost, represented by a list `cost` of length `n`, where `cost[i]` is the cost of stepping on step `i`. The goal is to find the minimum cost to reach the top of the staircase, starting from the bottom step (0). You can start by taking either 1, 2, or 3 steps. Write a function in Python which takes in the list `cost`, and returns a list that corresponds to the least-cost plan. The list could be for example

$$2, 1, 3$$

, which would correspond to "first take two steps, then one step, then three steps."

CONT'D...

Question 10. [10 MARKS]

We can use a dictionary to record who is friends with whom by recording the lists of friends in a dictionary. For example:

```
friends = {"Carl Gauss": ["Isaac Newton", "Gottfried Leibniz", "Charles Babbage"],
"Gottfried Leibniz": ["Carl Gauss"],
"Isaac Newton": ["Carl Gauss", "Charles Babbage"],
"Ada Lovelace": ["Charles Babbage", "Michael Faraday"],
"Charles Babbage": ["Isaac Newton", "Carl Gauss", "Ada Lovelace"],
"Michael Faraday": ["Ada Lovelace"] }
```

Here, Carl Gauss is friends with Isaac Newton, Gottfried Leibniz, and Charles Babbage. Assume that friendships are symmetric, so that if X is friends with Y, then it's guaranteed that Y is friends with X.

A *friendship chain* is a chain of people who are connected by friendship, with no repetitions allowed. For example, the following is a friendship chain of length 5:

Carl Gauss → Isaac Newton → Charles Babbage → Ada Lovelace → Michael Faraday

Write a function which takes in a dictionary in the format above, and returns the length of the longest friendship chain in the data in the dictionary.

You **must not** use recursion.

CONT'D...

Extra space for solutions

CONT'D...

Extra space for solutions

Extra space for solutions

CONT'D...

Extra space for solutions

Extra space for solutions

CONT'D...

Extra space for solutions

PLEASE WRITE NOTHING ON THIS PAGE