UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING

APS 105 — Computer Fundamentals
Final Examination
April 20, 2020

Examiners: P. Anderson, B. Li, B. Korst, M. Rezanejad

Exam Type D: This is an "open book" examination.

Please see the posted examination instructions in Quercus. *If the question calls for you to upload a **function** as your .c code, only include the function that needs to be implemented and your own helper functions. Do not include main or any sample data. When "planning material" is specified, this means your background program development information like the data structure drawings and flow charts you might use to analyse the problem. Your planning material is optional, and your .c code file is your primary submission for marks and part marks.*

You must use the C programming language to answer programming questions. Do not write `#include` directives in your solutions. You may use functions from the `math` library as necessary.

The examination has 18 pages, including this one. If you use the extra page(s) note this on the original question page.

First Name: _____ Last Name: _____

Your Student Number: _____

MARKS

1	2	3	4	5	6	7	8	9	10	Total
/1	/8	/4	/8	/8	/14	/14	/14	/14	/15	/100

Question 1 [1 Marks]

Please answer this question by filling in your name below to verify that you will not cheat. Note that we will be using a code checker (as we did for the labs), monitoring online, reading emails from students that report cheating behaviour from others, and using other methods to protect our students who are writing this exam in good faith.

Marks will be withheld from those students who do not get 1/1 on this question while we conduct a further investigation.

I, _____, pledge upon my honour that I will not violate our Faculty's Code of Behaviour on Academic Matters during this assessment by acting in any way that would constitute cheating, misrepresentation, or unfairness, including but not limited to, using unauthorized aids and assistance, impersonating another person, and committing plagiarism **AND** will not violate the Code of Behaviour on Academic Matters by providing unauthorized aids and assistance or impersonating another person since doing so is also considered a serious academic offence.

Question 2 [8 Marks]

Short Answer Questions

Submit these answers through an image or a PDF document, with question numbers and answers on it.

Question 2(a) (2 marks)

You have an array of string pointers, `pInstrings[10]`, which must be pointed to input strings of unknown length from a user. The function `getInstring()` returns a pointer to the string, which you must copy into dynamically-allocated memory. Complete the `for` loop in the code segment below, which calls `getInstring()` and copies the returned string to a new memory location. Assume that the string returned by `getInstring()` will persist until the next call to `getInstring()`.

```
for(int i = 0; i < 10; i++) {
    // put code here to call getInstring(), copy returned string to
    // dynamically-allocated space, then assign the address to pInstring[i]




}
```

Question 2(b) (2 marks)

```
int q[7] = {0, 3, 6, 9, 12, 15, 18};
int *b = &q[3];
int **a = &b;
```

What is `*(&(**a)+2)`?

(If it is an integer, specify the integer; if it is a pointer, specify what it points to; if it is an address, specify what is at that address.)

Question 2(c) (2 marks)

We want to count the number of the small letter `'e'` that appear in a string. This program does not work for several reasons. Correct all the mistakes in this C program so it works correctly. Assume that the string library is available.

```
void counte(string *instr) {
    for (int i = 1; i < strlen(instr); i++) {
        if (instr[i] = 'e') int count++;
    }

    return count;
}
```

Question 2(d) (2 marks)

A user writes as part of a program:

```
mammal tiger;
tiger.numFeet = 4;
```

Write a structure definition and any other statements that could be in place to make this compile correctly. You do not have to specify any include statements that include header files.

Question 3 [4 Marks]

Multiple Choice Questions

Submit these answers through an image or a PDF document, with question numbers and answers on it.

Question 3(a) (1 mark)

Assume that a binary search tree generated for the input sequence `-33, 45, -19, 2234, -1100` is traversed in left subtree, right subtree, root order, and that the data is printed at each node. What is the sequence of numbers that would be produced?

(a) `-1100 -33 -19 45 2234`

(b) `-33 45 -19 2234 -1100`

(c) `-1100 -19 2234 45 -33`

(d) `-33 -1100 45 -19 2234`

(e) `2234 45 -19 -33 -1100`

Question 3(b) (2 marks)

For which sorting method(s) studied in class are both of the following statements true?

1. After N passes, exactly N values are known to be in their final positions.
2. A total of exactly N-1 passes is required to complete the sort of N items.

(a) Bubble and insertion sort

(b) Bubble sort only

(c) Selection sort only

(d) Insertion and selection sort

(e) Bubble and selection sort

Question 3(c) (1 mark)

The following function is intended to define a recursive insertion sort algorithm that sorts the array **A** into ascending order. The recursive call is missing.

```
void insertion_sort(int list[], int size) {
    int i;
    if (size <= 1) {
      return;
    }

    /***** missing line *****/

    for (i = size - 1; i; --i) {
        if (list[i] < list[i - 1]) {
            swap(&list[i], &list[i - 1]);
        } else {
            break;
        }
    }
}
```

What should replace the missing line so that the algorithm is correct?

(a) `insertion_sort(list + 1, size - 1);`

(b) `insertion_sort(list, size);`

(c) `insertion_sort(list + 1, size);`

(d) `insertion_sort(list, size - 1);`

(e) `The recursive call should appear after the for loop`

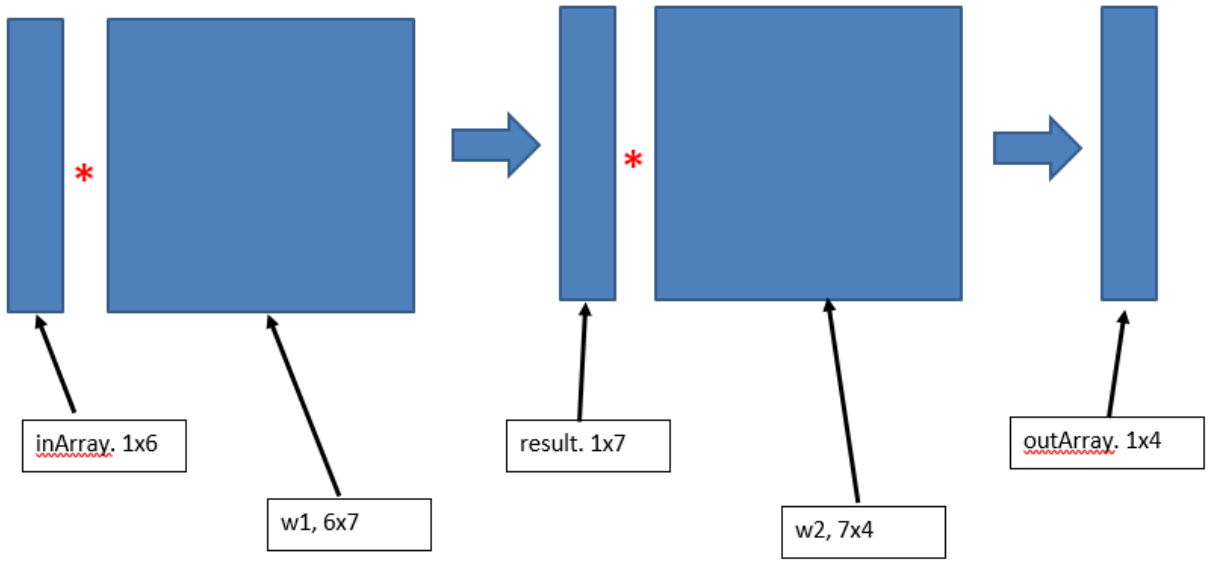
Question 4 [8 Marks]

Submit a .c code file with your solution, and optionally a file of your planning material (image or PDF).

For artificial intelligence (AI) that uses recently developed "neural network" techniques, layers of "neurons" pass their values, weighted by multipliers, into the next layers. Finding those weights is, of course, the more difficult and computationally intensive part, but once you have them the use is relatively simple array multiplication.

Here we will work with two layers to do the last stages and to compute values for the output. For example, the output could be the likelihood that the inputs are from the picture of one of four animals: dog, cat, pig, beaver. This net will "recognize" which of these it is by choosing the output of highest value.

Here is a diagram that represents what we are doing:



You will generate the function to perform the operation shown starting with the code below that includes the W1 and W2 weight arrays and that takes an input array. The input is one-dimensional, size 6. W1 and W2 are two-dimensional: W1 is 6 by 7. W2 is 7 by 4. The output is one-dimensional and of size 4.

```
Note: To multiply a Nx1 array A, by an MxN array B, to get an Mx1 array C:

C[0] = A[0] * B[0,0] + A[1] * B[0,1] + A[2] * B[0,2] + ... + A[N-1] * B[0,N-1]
C[1] = A[0] * B[1,0] + A[1] * B[1,1] + A[2] * B[1,2] + ... + A[N-1] * B[1,N-1]
...
C[M-1] = A[0] * B[M-1, 0] + A[1] * B[M-1, 1] + A[2] * B[M-1, 2]+ ... + A[N-1] * B[M-1, N-1]
```

Here is the function to finish:

```
#include <stdio.h>

// this function uses a one-dimensional input array of size 6, and
// puts values into the one-dimensional output array of size 4

void performAI(double *inArray, double *outArray) {
    // these are the weighting arrays as in the diagram
    double w1[6][7] = {
     {0.795279571, 0.565454091, 0.569392801, 0.649519912, 0.311228459, 0.869033219, 0.963890145},
     {0.261182548, 0.967901447, 0.015463096, 0.101966965, 0.454071297, 0.396147575, 0.853833996},
     {0.976180547, 0.762522649, 0.223067359, 0.120228416, 0.710471648, 0.220771538, 0.052876278},
     {0.173285965, 0.795507616, 0.258332188, 0.813302777, 0.528470338, 0.885245811, 0.190564347},
     {0.14018923, 0.324797853, 0.012649753, 0.928397252, 0.048519668, 0.321836138, 0.360198988},
     {0.063248883, 0.72395506, 0.606492812, 0.435057638, 0.462896967, 0.12061378, 0.28806367}};

    double w2[7][4] = {
        {0.036340161, 0.702081192, 0.406643568, 0.383400727},
        {0.786459022, 0.627286192, 0.190417846, 0.259622675},
        {0.996272492, 0.115783107, 0.922042574, 0.805576672},
        {0.254649714, 0.818737484, 0.23760355, 0.884876231},
        {0.587934606, 0.566762923, 0.254228386, 0.735145224},
        {0.709219708, 0.815306359, 0.395073347, 0.191438772},
        {0.743663242, 0.969784133, 0.055612809, 0.992284824}};
    // your code here


};
```

Here is an example use of the function. *Do NOT include the* `main()` *function in your submitted file*:

```
    double inData[] = {10, 11, 14, 51, 22, 24};
    double outData[4];

    performAI(inData, outData);
```

Question 5 [8 Marks]

Submit a .c code file with your solution, and optionally a file of your planning material (image or PDF).

An integer with the type `unsigned int` has 32 bits, and stores values from 0 to $2^{32} - 1$. Write a function `unsigned int generateInt(int list[], int size)`, that takes a list of *distinct* integers valued from 0 to 31 as its first parameter, and the size of such a list as its second parameter, and returns an integer with the corresponding bits set as 1.

For example, if the input list is the array `[0, 3, 4, 9]` with a `size` of 4, the function will return `1000011001` in binary (base-2) form, as bit 0, bit 3, bit 4, and bit 9 (counting from the least significant bit on the right) are set to `1`. The following `main()` function will print `result = 219` as the result `1000011001` is printed in hexadecimal form by using the format specifier `%x`:

```
int main(void) {
    int list[] = {0, 3, 4, 9};

    unsigned int result = generateInt(list, 4);

    printf("result = %x\n", result);
    return 0;
}


unsigned int generateInt(int list[], int size) {
  // your code here









}
```

Question 6 [14 Marks]

Submit a .c code file with your solution, and optionally a file of your planning material (image or PDF).

You are asked to implement a function `void convert(char *input, char *output, int lineLength)` to convert the long string `input` to a string to be placed in the character buffer `output`, so that each line contains as many words as possible, but is no longer than `lineLength` characters. The newline character, `'\n'`, is used to separate the lines, and the `convert()` function should insert or remove newline characters where needed. The result of this conversion should not end a line or start a new line with a blank space (`' '`).

You may assume that the output buffer `output` has sufficient space for the string after conversion, and that no single word in the `input` string is longer than `lineLength`.

For example, if we use the following `main()` function:

```
int main(void) {
  char *input =
    "I'd like to believe that this works just fine.\nBut then again it might not.\n";
  char output[2048];

  convert(input, output, 15);

  printf("%s\n", output);

  // Making all spaces visible by replacing them with 'X'
  for(int i = 0; i <= strlen(output); i++) {
    if (output[i] == ' ') {
      output[i] = 'X';
    }
  }
  printf("%s\n", output);
  return 0;
}
```

It will produce the following result:

```
I'd like to
believe that
this works just
fine.

But then again
it might not.

I'dXlikeXto
believeXthat
```

```
thisXworksXjust
fine.

ButXthenXagain
itXmightXnot.


void convert(char *input, char *output, int lineLength) {
  // write your code here









}
```

Question 7 [14 Marks]

Submit a .c code file with your solution, and optionally a file of your planning material (image or PDF).

Complete the definition of a C function called `removeStringFromString()`, whose prototype is shown below, that takes two parameters, a string `str` and another string `substr`, and returns a new string that contains a copy of `str`, but with *all* instances of `substr` removed. For example, the following code in the `main` function:

```
char *s = "Hello World!";
const char *p = removeStringFromString(s, "l");
const char *q = removeStringFromString(s, "el");
printf("%s\n", p);
printf("%s\n", q);
```

would print:

```
Heo Word!
Hlo World!
```

You **must** use recursion to solve this problem.

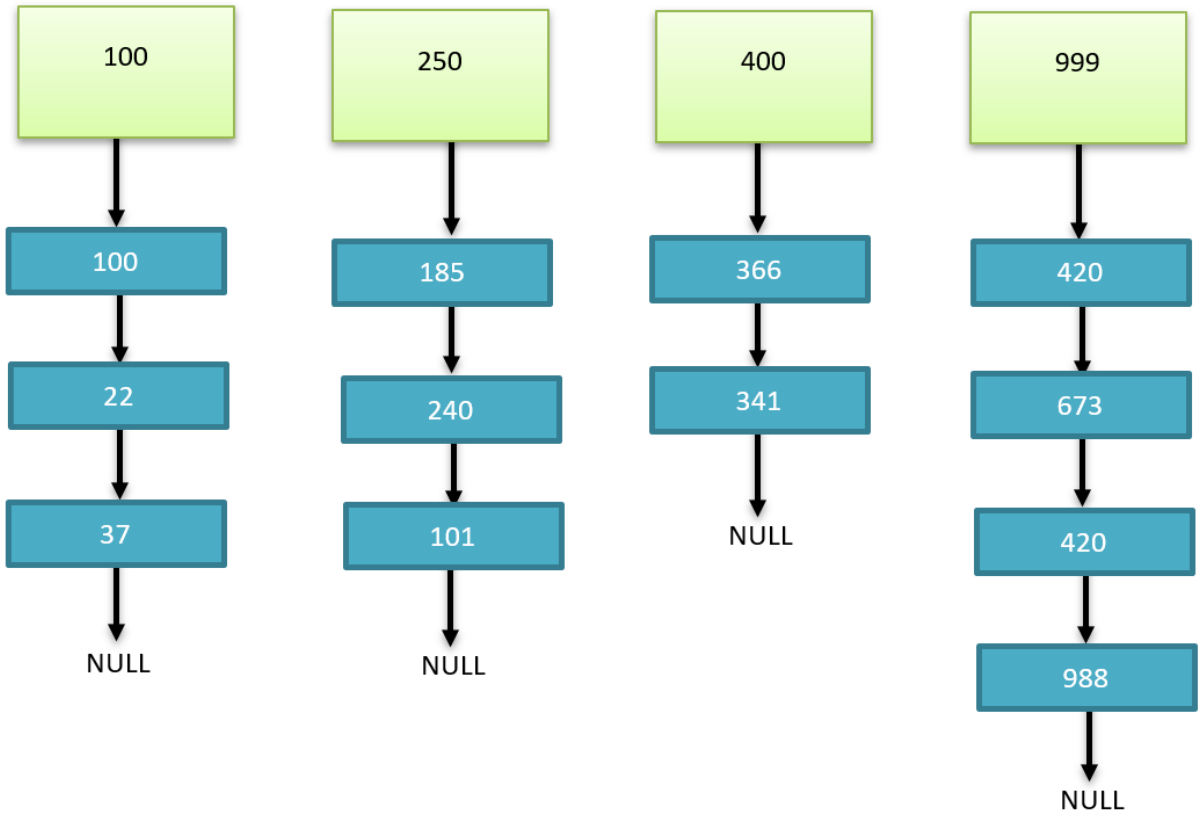
Hint: You may wish to define your own helper function.

```
const char *removeStringFromString(const char *str, const char * substr) {




}
```

Question 8 [14 Marks]

Submit a .c code file with your solution, and optionally a file of your planning material (image or PDF).

One way of sorting data for retrieval is to put it into groups within a range. To find something you go to the group with the right range, then do a linear search of the data in that group. We show an example in the figure below:



Here we have four groupings of non-negative numbers from 0 to 999. The first group is 0-100, the second from 101-250, the third from 251-400 and the last from 401 to 999. The value in each group shows the maximum allowed in that group, and each group carries on with values bigger than those that go in the previous group. So the second group carries on from the first (which only goes to 100) and goes to the maximum of that group which is 250.

Note that duplicates are allowed, and that the list in each group is not sorted. We will only deal with non-negative numbers. What you are going to do is to take the starting code below and add two functions:

- `listNode *insertGNode(int value)` that inserts value into the start of the list of the correct group, and returns a pointer to that new node or NULL if too big for any group. A reminder that duplicates are OK so you don't have to check as to whether the node already exists.
- `int foundGNode(int valIn)` that returns a 0 if valIn is in a node in some group, 1 otherwise.

You can test using the following environment. *Do NOT include these in the .c file that you submit*. Your functions must work with other values of NUMGROUPS and maxInGroup values in groups[]; the groups defined below are the same as in the figure.

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

// nodes of the list
struct listNode {
int value;
struct listNode *nextNode;
};

// group head
struct groupNode {
int maxInGroup;
struct listNode *firstNode;
};

#define NUMGROUPS 4

Struct groupNode groups[NUMGROUPS] = { {100, NULL}, {250, NULL}, {400, NULL}, {999, NULL} };

//Here are the functions for you to define:
// put the value in groups. Duplicates are OK. Return NULL if the value too big for any group
struct listNode *insertGNode(int value) {



  // put your code here


}

// see if valIn exists in groups.
int foundGNode(int valIn) {

    //put your code here

}
```

Question 9 [14 Marks]

Submit a .c code file with your solution, and optionally a file of your planning material (image or PDF).

Imagine we have a sorted array of `int`-type integers, a, in the following form:

$$a = \{a_1, a_2, \ldots, a_{N-1}, a_N\}$$

where $a_1 \leq a_2 \leq \ldots \leq a_{N-1} \leq a_N$. We now define the following array from the sorted array a:

$$a_d = \{a_1, a_N, a_2, a_{N-1}, \ldots\}$$

In this array a_d, the smallest item is followed by the greatest item, and then the second smallest item and the second greatest item, and so on. We call this special array a Decussate-sorted array. You are asked to write a function `int *merge(int *size)`, which receives a series of Decussate-sorted arrays of different lengths from user input, and returns a single-dimensional array that merges them all together to one Decussate-sorted array. The `merge()` function has one parameter `size`, which is used to return the size of the merged array to the calling function. Your merge function is responsible for getting the array information from the user in the manner shown below.

Note: Your implementation should have no *memory leaks*. In other words, any dynamically allocated memory that you use inside the function should be freed. The returned array from the `merge()` function will be freed by the calling function.

Hint: You may use a sorting algorithm if that simplifies your solution.

Here is an example `main()` function that can be used to test your work:

```
int main(void) {
    int size;

    int *mergedArray = merge(&size);

    printf("Result: ");
    for (int i = 0; i < size; i ++) {
        printf("%d ", mergedArray[i]);
    }
    printf("\n");

    free(mergedArray);
    return 0;
}
```

And here is an example run of this program:

```
Please enter the number of arrays you have: 3
Please enter the size of array number 1: 4
Please enter the array number 1: 12 78 23 45
Please enter the size of array number 2: 3
```

```
Please enter the array number 2: 10 28 14
Please enter the size of array number 3: 5
Please enter the array number 3: 17 48 22 36 25
Result: 10 78 12 48 14 45 17 36 22 28 23 25

int *merge(int *size) {









}
```

Question 10 [15 Marks]

Submit a .c code file with your solution, and optionally a file of your planning material (image or PDF).

Do **not** *include the provided structure definitions in your .c file.*

The *height* of a binary search tree is defined as the number of levels between the root and the leaves of the tree (including the root, so a tree with a single node has height = 1). Given a binary search tree, write a C function called `isTreeSkewed()`, whose prototype is shown below, that takes a binary search tree as its first parameter, the number of top levels considered n as its second parameter, and returns a boolean value to indicate whether the tree is skewed in height or not. A binary search tree is *skewed* if the left and right subtrees of every node in the top n levels of the tree differ in height by greater than 2.

For example, running the `main()` function provided to you in the sample code for this question, it will print the following:

```
The tree contains (in ascending order): 1 2 3 4 6 7 8 9 15 18 30 31 32
The tree is not skewed.
```

```
typedef struct node {
  int data;
  struct node *left, *right;
} Node;

typedef struct bstree {
    Node *root;
} BSTree;

bool isTreeSkewed(BSTree *tree, int n) {









}
```

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