CIS*2520— Assignment #1

Fall 2024

Due: Friday, September 27, 2024@ 23:59

Please submit your assignment solutions as **one PDF file** (named as YOUR/UoG/ID_a1.pdf, e.g. 1234567_a1.pdf) to Dropbox under Assignment 1 before the due date.

You are granted a penalty-free grace period for 48-hours. The grace period ends on Sunday, September **29** 23:59. After this time, you cannot submit the assignment. The late assignment (no submission after the grace period) will be marked as ZERO.

If you require a longer grace period (than the default 48 hours) for your assignment, the request must be sent to the course email: cis2520@socs.uoguelph.ca (using your UoG account) BEFORE the due date (Friday September 27, 2024@ 23:59). Please refer to the communication guidelines in week 0 on writing emails.

Please refer to the Course Outline and Academic Integrity Video to ensure you understand and comply with the University's Academic Integrity Standards.

This assignment covers the course content in weeks 0 – 2. Total 40 points.

1. C programming function calls (4pts)

1) There are two ways to the arguments can be passed during the function calls in C, call by value and call by reference. In your own words, describe the differences between the two (in 2-3 sentences).

2) Read the follow program. Using your response in 1), explain which variable is passed by value, and which one is passed by reference.

Note: In 2), you MUST use your response in 1) to explain the answer.

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

```
// Function 1
void fun1(int a) {
    a = a + 1;
}

// Function 2
void fun2(int *b) {
    *b = *b + 1;
}

int main() {
    int x = 10;
    int y = 10;

    printf("Before the function call: x = %d\n", x);
    fun1(x);
    printf("After the function call: x = %d\n\n", x);

    printf("Before the function call: y = %d\n", y);
    fun2(&y);
    printf("After the function call: y = %d\n", y);

    return 0;
}
```

2. C programming pointers (12pts)

There are some errors in the following program. Please identify them and provide the correct statements.

Note: Some lines of code with errors may be compiled and executed but will not do what it is expected to do.

```
int main() {

   int A[5]={1,3,7,4,0};
   int *P[5], *pA;

   int i, x, y;

//Set pointer to the base address of array
*pA = &A;

// Assign A[1] to x and y
x = *pA+1;
A++;
y=*A;


printf("x = \t%d",x);
printf("y = \t%d",y);


   //Set every pointer to one array element

   for(i=0;i<5;i++)

     *P[i]=&A[i];

   //print array elements using pointers

   for(i=0;i<5;i++)

     printf("\t%d",P[i]);

   return 0;

}
```

3. Big-O, big Omega, and Theta (9pts)

Referring to the definitions and examples of Big-O, big Omega, and Theta introduced in the course slides, show that

1) $3n^2 = O(n^3)$

2) $n^2+2n \neq \Omega(n^3)$
3) $n^2-2n = \boldsymbol{\Theta}(n^2)$

Note: you MUST use the definitions of the different notions introduced in class to solve the questions.

4. Bubble sort and revised bubble sort total operations (12pts)

Bubble sort is a simple sorting algorithm. It compares each adjacent pair and check if the elements are in order. If they aren't, the elements are swapped. This process continues until all elements are sorted.

If there is an integer array of *n* elements, and the algorithm output the array in an increasing order (from smallest to the largest), the bubble sort can be implemented as

```
for(int i = 0;i < n; i++){
    for(int j=0;j < n - 1; j++){
        if(arr[j] > arr[j+1]){
            int temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp;
        }
    }
}
```

You have an array X=[1,2,3,…,n] and an array Y=[n,n-1,n-2,…,1]. So, X represents the best case (the array is already sorted) and Y represents the worst case (the array is sorted reversely).

1) Analyze the above bubble sort and calculate the total number of operations with input X and Y, respectively.

There is another way of implementing bubble sort where the algorithm also checks if the array has been sorted (to improve the efficiency). We create a flag that checks if a swap has occurred between any adjacent pairs. If there is no swap while traversing the entire array, the array is completely sorted, and the algorithm can break out of the loop. This is called a revised bubble sort. It can be implemented as

```
for(int i = 0;i < n; i++){
    boolean isSwapped = false;
    for(int j=0;j < n - 1; j++){
       if(arr[j] > arr[j+1]){
            int temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp;
            isSwapped = true;
        }
     }
    if(!isSwapped){
        break;
   }
}
```

2) Analyze the above revised bubble sort and calculate the total number of operations with input X and Y, respectively.

Note: you MUST include the calculation steps and explain how you get the answers.

5. Complexity (3pts)

Suppose you have a computer that requires 1 minute to solve problem instances of size n = 100. What instance sizes can be run in 1 minute if you buy a new computer that runs 64 times faster than the old one, assuming the Time complexities T (n) ∈ $\boldsymbol{\Theta}$ *(n^2)* for the algorithm?

Note: you MUST include the calculation steps in your answer.