CMPT 125, Fall 2019

Midterm Exam
October 28, 2019

Name_____

SFU ID: |__|__|__|__|__|__|__|__|__|

Problem 1	
Problem 2	
Problem 3	
Problem 4	
TOTAL	

Instructions:

1. Duration of the exam is 90 minutes.
2. Write your name and SFU ID **clearly**.
3. This is a closed book exam, no calculators, cell phones, or any other material.
4. The exam consists of four (4) problems. Each problem is worth 25 points.
5. Write your answers in the provided space.
6. There is an extra page at the end of the exam. You may use it if needed.
7. Explain all your answers.
8. Really, explain all your answers.

Good luck!

Problem 1 [25 points]

a) [6 points] What will be the output of the following program?

```
#include <stdio.h>
enum week {MON, TUE, WED, THUR, FRI, SAT, SUN};

void foo(int* x, int y, int* z) {
    x = z;
    y = *x;
    *z = THUR;
}

int main() {
    int a = MON, b = TUE, c = WED;
    foo(&a, b, &c);
    printf("a = %d, b = %d, c = %d", a, b, c);
    return 0;
}
```

b) [4 points] Will the code below compile?
If yes, what will be the output? If no, explain why.

```
#include <stdio.h>

int main() {
  char str[5] ={'a','b','c','d',0};
  char* ptr = str;
  printf("%s\n", ptr);
  return 0;
}
```

c) Consider the following function.

```
int bar(int n) {
  if (n <= 0)
    return 0;
  else {
    int i = 0, sum = 0;
    while(sum <= n*(n-1)/2) {
      i++;
      sum += i;
    }
    return i + bar(n-1);
  }
}
```

**Below you may need the following fact: 1+2+3+...+n = n*(n+1)/2.

[4 points] What does bar(n) return on input n = 3? Show some intermediate computation if needed.

[5 points] Use the big-O notation to express the running time of bar(n) as a function of n. Explain your answer.

[6 points] Explain in words what bar(n) returns, and write a function with the same functionality as bar(n) whose running time is O(1).

Problem 2 [25 points]

a) [15 points] Recall the MergeSort algorithm.

```
void merge_sort(int* A, int n) {
  if (n >= 2) {
      int mid = n/2;
      merge_sort(A, mid);
      merge_sort(A+mid, n-mid);
      merge(A,n,mid);
  }
}
```

Implement the merge function that gets an array A of length n, and an index mid, and it is guaranteed that the part A[0,...mid-1] is sorted in ascending order, and A[mid...n-1] is sorted in ascending order. The function merges the two halves in time in A in time O(n).
You may assume all elements are distinct.
Remember to use malloc/free if you need to use a new array.

```
void merge(int* A, int n, int mid) {









}
```

b) [6 points] Consider the ***QuickSort*** algorithm that uses as a pivot the first element (i.e., A[0]). List all recursive calls made by the algorithm on input A = [2, 8, 6, 1, 5, 3].
Show some intermediate steps of the computation.

c) [4 points] Consider the **Binary Search** algorithm. How many comparisons will it make on input A = [2, 4, 6, 8, 10, 12, 14] when searching for 6.

Problem 3 [25 points]

a) [6 points] Consider the following sequence of operations on a stack. What will be the state of the stack in the end? Show some intermediate steps of the computation.

```
stack_t* s = stack_create();
push(s, 1);
push(s, 2);
push(s, 3);
push(s, 4);
pop(s);

push(s, 5);
push(s, 1);
pop(s);
push(s, 1);
push(s, 3);
pop(s);
```

b) [6 points] Consider the following sequence of operations on a queue. What will be the state of the queue in the end? Show some intermediate steps of the computation.

```
queue_t* q = queue_create();
enqueue(q, 1);
enqueue(q, 2);
enqueue(q, 3);
enqueue(q, 4);
dequeue(q);

enqueue(q, 5);
enqueue(q, 1);
dequeue(q);

enqueue(q, 1);
enqueue(q, 3);
dequeue(q);
```

c) Consider the following function.

```
void do_what(queue_t* q) {
   // Base case
   if (queue_is_empty(q));
       return;
   // Dequeue current item (from the head)
   int data = dequeue(q);
   // apply recursion
   do_what(q);
   // Enqueue current item (to the tail)
   enqueue(q, data);
}
```

[8 points] If the input to the function is a queue in the state q = [1,2,3] (1 is the head, 3 is the tail), what will be the state of q when the function returns? Explain your answer.

[5 points] Use the big-O notation to express the running time of do_what. Explain your answer.

Problem 4 [25 points]

In this problem we represent a *Linked List of ints* using `LLnode_t`:

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

a) Consider the following function

```
int fun_list(LLnode_t* head)  {
  if (head == NULL) {
     printf("\n");
     return 1;
  }
  else  {
    int w = fun_list(head->next);
    printf("%d  ", head->data);
    return w * head->data;
  }
}
```

[4 points] What will fun_list() *return* on input 1→ 2→ 3→ 2→ 5? Explain your answer.

[4 points] What will fun_list() *print* on input 1→ 2→ 3→ 2→ 5? Explain your answer.

b) [12 points] Write a function in C that gets a linked list and a number, and returns the first node containing the number. If the number is not in the list, the function returns `NULL`.
The linked list is represented as a pointer to the first node in the list `LLnode_t* head`.

The state of the list on return should not change.

```
LLnode_t* find(LLnode_t* head, int number) {









}
```

[5 points] Use big-O notation to express the running time of your function?

Extra page