CMPT 125, Spring 2022

Final Exam
April 26, 2022

Name________________________

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

Problem 1	
Problem 2	
Problem 3	
Problem 4	
TOTAL	

Instructions:

1. Duration of the exam is 180 minutes.
2. Write your full 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 if the question asks you to.
8. Really, explain all your answers.

Good luck!

Problem 1 [25 points]

a) [6 points] What will be the output of the following code? Explain your answer.

```
#include <stdio.h>
int what(int n) {
    if (n<=0)   return 0;
    int sum = 0;
    for (int i=1; i<=n; i=i+2) {
      printf("%d ", i);
      sum += i;
    }
    printf("\n");
    return sum + what(n/2);
}
int main(void) {
     printf("what(6) = %d", what(6));
     return 0;
}
```

b) [6 points] Will the code below compile?
If yes, describe what the code does? If not, explain why.

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

char* bar() {
  char* str = malloc(2);
  str[0] = 'A';
  char* ret = str+1;
  ret[1] = 'B';
  return ret;
}
int main() {
    char* a = bar();
    printf("%s\n", a);
    free(a);
    return 0;
}
```

c) [8 points] A string is said to be a good password if it satisfies the following conditions:

- its length is at least 8
- contains at least one digit
- contains at least one lower case letter
- contains at least one upper case letter
- contains at least one of these three symbols: !@#

Write a function that gets a string, and checks if it is a good password.

```
#include <stdio.h>
bool is_good_password(const char* str) {

}
```

d) [5 points] Explain the differences in C++ between the following:

- passing a variable to a function by value.
- passing a variable to a function using a pointer.
- passing a variable to a function using a reference.

Explain how each of them can be used. Provide examples if that helps the explanation.

```
}
```

Problem 2 [25 points]

An array A[0...n-1] of ints is called *a mountain* if there exists an index 0≤k≤n-1 such that

- arr[0] < arr[1] < ... < arr[k-1] < arr[k]
- arr[k] > arr[k+1] > ... > arr[n-1]

The index k is called the peak.
Note that if k=0 then A is a decreasing array, and k=n-1 corresponds to an increasing array.

a) [10 points] Write a function in C that solves the following problem.
Input: A mountain array of ints A of length n>0 with all values distinct
Output: the index of the peak.
The running time: must be O(log(n)).
Examples:

- For A = [1, 4, 5, 6, 7, 3, 1], the output should be 4 (because, A[4] = 7).
- For A = [-5, 1, 3], the output should be 2 (because A[2] = 3).
- For A = [10, 4], the output should be 0 (because A[0] = 10).

Explain your idea before writing code.

```c
int find_peak(const int* A, int n) {






}
```

b) [15 points] Write a function that gets a mountain array of length n, and the index k representing the peak, and sorts the array in the increasing order in O(n) time.
Explain your idea before writing code.

```
void sort_mountain(int* A, int n, int k) {









}
```

Problem 3 [25 points]

In the problem a linked list of ints is represented as follows.

```
struct LL_node {
  int data;
  struct LL_node* next;
};
typedef struct LL_node LL_node_t;

typedef struct {
  LL_node_t* head;
  LL_node_t* tail;
} LL_t;
```

a) [13 points] Write a function in C that gets a linked list of ints and checks if all negative numbers in the list come before all non-negative numbers. (zero is non-negative)
The running time of the function must be O(length of list).

For example, the function returns true on the following inputs:

-2 → -4 → 10 → 0 → 11
4 → 3 → 5
-2 → -4
1
-6
empty list

The function needs to return false on the following inputs:

-3 → 4 → 6 → -8
7 → -1
1 → 0 → -1 → 1

```
bool negative_first(LL_t* list) {




}
```

b) [12 points: 6 points each] Implement the following standard functions on a linked list with pointers to head and tail, using the struct above. Each of the functions must run in O(1) time.

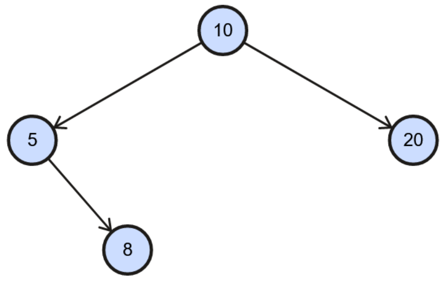
```
// adds a node with the given value to the end of the list
void add_to_tail(LL_t* list, int val) {















}

// removes the first node from the list, and returns its value
// assumption: list is not empty
int remove_from_head(LL_t* list) {















}
```

Problem 4 [25 points]

In this problem use the following struct for Binary Tree of ints.

```
struct BTnode {
  int value;
  struct BTnode* left;
  struct BTnode* right;
  struct BTnode* parent;
};
typedef struct BTnode BTnode_t;
```

a) Consider the following Binary Search Tree.



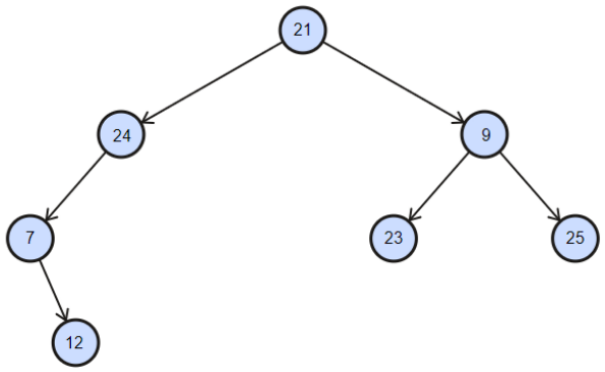
[3 points] Add the list of numbers to the Binary Search Tree in the given order: 4, 22, 21, 23. What will be the result in the end? Draw the resulting tree with the eight nodes.

[4 points] Find four more permutations (reorderings) of the numbers {4,21,22,23}, such that for each of the permutations if we add the numbers in this order to the original BST, the result will be the same as in your answer above. Explain your answer.

[3 points] Remove 10 from the BST obtained above (having the values 4,5,8,10,20,21,22,23). What will be the result?

b) [15 points] Write a function that gets a node in a binary tree (not necessarily the root), and returns the next node in the *in-order traversal order*.
The running time must be O(depth of three).
For example, in the tree below:

- on input 7 the output should be 12
- on input 24 the output should be 21
- on input 21 the output should be 23
- on input 25 the output should be NULL



Explain your answer before writing the code.

```
BTnode_t* next_inorder(BTnode_t* node) {







}
```

Extra page

Empty page

Empty page