CMPT 125, Fall 2022

Midterm Exam
October 24, 2022

Name________________________

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

Problem 1	
Problem 2	
Problem 3	
Problem 4	
TOTAL	

Instructions:

1. Duration of the exam is 100 minutes.
2. Write your full name and SFU ID NUMBER **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] Will the program compile? If yes, what will be the output of the following program? If not, explain why.

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

void do_something(int* a, int n) {
     a = malloc(n*sizeof(int));
     for(int i=0; i<n; i++)
          a[i] = i*i;
}

int main() {
     int* ar;
     do_something(ar, 5);
     for(int i=0; i<5; i++)
          printf("a[%d] = %d\n", i, ar[i]);
     free(ar);
     return 0;
}
```

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

```c
#include <stdio.h>

int main() {
    char s[] = {'a','b',0,'1','2','3','4',0,'x','y','z'};
    int ind=0;
    while (s[ind])
      ind = ind+1;
    printf("%s\n", s+ind+2);
    return 0;
}
```

For c) and d) consider the following function

```
#include <stdio.h>

long what(int n) {
  if (n==0)
    return 1;
  return what(n-1)+what(n-1);
}
```

c) [7 points] Explain in words the functionality of the function. Given examples of input-output pairs for this function, e.g., what will be the output on input 3,4,5?

d) [6 points] Rewrite the function without using recursion, so that it has the same functionality, and running time O(n).

Problem 2 [25 points]

a) [5 points] Consider the **Binary Search** algorithm. List all comparisons it makes on the input A = [2, 4, 6, 8, 9, 12, 14, 15, 18, 90, 99] when searching for 70. Explain your answer.

b) [8 points] Show an array with the values {1,2,3,4,5,6,7,8} so that **Insertion Sort** makes:

- exactly 1 swaps in the first iteration of the outer loop,
- exactly 7 swaps in the last iteration of the outer loop,
- no swaps in other iterations.

(Since the array has 8 elements, the outer loop of Insertion Sort has 7 iterations)
Explain your answer.

c) [12 points] Implement a recursive version of Binary Search. The function gets an array A of length n and an element elt.

- If elt is in A, the function returns an index i such that A[i]==elt.
- If elt is not in A, the function returns -1.

You need to write a recursive implementation.
Explain your code if necessary.

```
int binary_search_rec(int* A, int n, int elt) {

}
```

Problem 3 [25 points]

a) [20 points] *Write a function that gets an array of length n>0 of strings containing non-negative integers, and returns the index of the largest one. If there are two maximal elements, the function can return the index of any of them.*

```
typedef const char* const_str; // define a type for const string
int str_num_max(const_str const numbers[], int n);
```

For example,

`str_num_max(["9","0","1","0"],4)` *returns 0, because 9 is the largest.*
`str_num_max(["5","8","2","48","3","48","0","18"],8)` *returns 3 or 5.*
`str_num_max(["52","52","52","52","52","52"],6)` *returns any number 0…5.*
`str_num_max(["934"],1)` *returns 0.*
`str_num_max(["214378798","54190238674879806948","8"],3)` *returns 1.*
`str_num_max(["98071523454909128474927382 9","511","9","561"],4)` *returns 0.*

1. *You may assume that the input is always legal, i.e., the strings represent positive integers in the correct format (no unnecessary leading zeros), all numbers are distinct, and n>0.*
2. *Note that the numbers may be larger than the maximum of* `int` *or* `long`.
3. *You may use standard library functions, e.g., functions from string.h.*
4. *You may write helper functions if needed. (If this hint is not clear, it means, you should definitely write helper functions to make your code look clean and readable)*

b) [5 points] Use big-O notation to analyze the running time of your function. Explain your answer.

Problem 4 [25 points]

a) [15 points] Write a function that gets a string and removes from it all spaces (ascii code 32).

```
void remove_spaces(char *str);
```

For example, consider the following code.

```
  char str[] = " ab  cd*1_2 @ ";
  remove_spaces(str)
  printf("%s", str);
```

It will print `"abcd*1_2@"`.

For full marks your function needs to run in time O(strlen(str)) and use O(1) extra space.

b) [7 points] Write a function that gets a string and checks if it is a palindrome of odd length.

```
bool is_odd_palindrome(const char* str);
```

For example:

- *On input* `"a2z@d@z2a"` *the function returns* `true`*.*
- *On input* `"ae11&edwdccek"` *the function returns* `false`*.*
- *On input* `"abccba"` *the function returns* `false`*, since the length is not* ***odd****.*

[3 points] What is the running time of your function? Use big-O notation to state your answer. Give the tightest possible answer. Explain your answer.

Extra page

Empty page