CMPT 125, Fall 2020

Midterm Exam - Solutions
October 26, 2020

Name_____

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

Problem 1	
Problem 2	
Problem 3	
Problem 4	
TOTAL	

Instructions:

1. Duration of the exam is 90 minutes.
   You have extra 30 minutes to submit your solution to Coursys.

2. You should write your solutions directly in this word file,
   and submit it to Coursys before 4:30pm.

3. Write your name and SFU ID on the top of this page.

4. This is an open book exam.
   You may use textbooks, calculators, wiki, stack overflow, geeksforgeeks, etc.
   If you do, specify the references in your solutions.

5. Discussions with other students are not allowed.
   Posting questions online asking for solutions is not allowed.

6. The exam consists of four (4) problems. Each problem is worth 25 points.
7. Write your answers in the provided space.

8. Explain all your answers.
9. Really, explain all your answers.

Good luck!

Problem 1 [25 points]

a) [5 points] Write a function that gets as input (1) the day of the week (MON, TUE,...) on October 1 and (2) a day of October, and returns the day of the week on that day?
See main() for testing examples.

```
#include <stdio.h>
typedef enum {MON, TUE, WED, THUR, FRI, SAT, SUN} weekday;
// assumption: day_of_month is between 1 and 31
weekday get_day(weekday oct1_day_of_week, int day_of_month) {
// implement me
```

Answer:

```
    return (oct1_day_of_week + day_of_month - 1) % 7;
}
int main() {
  printf("Testing: If Oct 1 is Monday, then Oct 7 is Sunday: ");
  if (get_day(MON, 7) == SUN)
    printf("ok\n");
  else
    printf("fail\n");

  printf("Testing: If Oct 1 is Thursday, then Oct 20 is Tuesday: ");
  if (get_day(THUR, 20) == TUE)
    printf("ok\n");
  else
    printf("fail\n");

  return 0;
}
```

b) [5 points] Will the code below compile?
If yes, what will be the output? Explain your answer. If no, explain why.

```
#include <stdio.h>
int main() {
  char str[10] ={'a','b','c',0,'1','2'};
  char* ptr = str;
  printf("%s\n", ptr+1);
  return 0;
}
```

ANSWER: will print “bc”.

Reason: ptr+1 points to ‘b’, and after ‘c’ null terminator.

c) Consider the following function.

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

**You may need the following fact: 1+2+3+...+i = i*(i+1)/2.

[4 points] What does bar(n) return on n = 10? Show intermediate steps of the computation.

Answer: In the beginning i=0, sum=0, 2sum-i = 0
After the first iteration we have: i=1, sum=1, 2sum-i = 1
After the second iteration we have: i=2, sum=3, 2sum-i = 5
After the third iteration we have: i=3, sum=6, 2sum-i = 9
After the fourth iteration we have: i=4, sum=10, 2sum-i = 16

The function stops and returns i=4.

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

Answer: In the i'th iteration sum=1+2+3+...+i = i*(i+1)/2
The while loop continues until 2sum-i=i^2 becomes more than n.
Therefore, the number of iterations is equal to sqrt(n).
In each iteration the running time is O(1).
Therefore, the total running time is O(sqrt(n))

[6 points] Explain in words what bar(n) returns.
Write a function with the same functionality as bar(n) whose running time is O(1).
If needed you may use sqrt(), ceil(), floor(), round() from math.h

Answer: For positive n the function returns sqrt(n) rounded up.

```
int bar(int n) {
 if (n<=0)
   return 0;
 else
   return (int)ceil(sqrt(n));
}
```

Problem 2 [25 points]

a) [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 15?

Answer: First compare to 8, then compare to 12, then to 14.
Total 3 comparisons

b) [4 points] Implement the function is_sorted() that gets an array A of length n, and checks if it is sorted in non-decreasing order.

```
bool is_sorted(int* A, int n){
 // implement me
```

Answer:

```
 for (int i=0; i<n-1; i++) {
   if (A[i] > A[i+1])
     return false;
 }
 return true;
}
```

[2 points] Use big-O notation to express the running time of your algorithm.

Answer: We have n-1 iterations in the worst case, with O(1) operations per iteration. Therefore, the total running time is O(n).

c) [6 points] Consider the **MergeSort** we saw in class with O(n) time merge procedure.
What is the running time of **MergeSort** on a sorted array of length n?
Use big-O notation to state your answer. Explain your answer.

Answer: The running time is O(n log(n)) on sorted arrays of length n.
The recursive calls do not depend on the values in the array, and for the merge procedure takes n/2 comparisons to merge two halves of length n/2 each.
Therefore, the total runtime is
$T(n) = 2T(n/2) + O(n) = O(n \log(n))$

d) [4 points] Consider the following variant of **MergeSort**:

```
// merging procedure we saw in class
void merge(int* A, int n, int mid);

// checks if the array is sorted
void is_sorted(int* A, int n);

void merge_sort(int* A, int n) {
  if (n <= 1 || is_sorted(A,n)) // stopping condition
    return;

  // recursion
  int mid = n/2;
  merge_sort(A, mid);
  merge_sort(A+mid, n-mid);
  merge(A,n,mid);
}
```

What is the running time of this algorithm when applied on A=[n/2+1,n/2+2,…,n,1,2,3...n/2]?

That is, the first half of A is [n/2+1,n/2+2,…,n], and the second half is [1,2,3,..,n/2].

Write the tightest possible upper bound on the running time. Explain your answer.

Answer:

1) The if() part runs in time O(n).
2) The recursive calls will run in O(n) time each since the two halves are sorted
3) Merge also takes O(n) time

Therefore, the total runtime is O(n)

e) [5 points] Consider the **QuickSort** algorithm that uses the first element (A[0]) as the pivot.

List all the **swaps** made by the algorithm (in all recursive calls) on input the A = [3, 2, 6, 0, 4]. Show the intermediate steps of the computation.

Answer:

pivot=3: **swap(6,0)** → [3,2,0,6,4]
The we **swap(3,0)** to move 3 to the right position
We make 2 recursive calls: [0,2] and [6,4]
[0,2] will have no swaps
[6,4] will have **swap(6,4)**

Problem 3 [25 points]

In the question below we use the following structs.
The structs represent information about students and their list of grades in all courses.

```
typedef struct {
  char* courseID;          // courseID, e.g. "CMPT125"
  int grade;               // numerical grade
} course_grade;

typedef struct {
  unsigned int ID;
  char* name;
  course_grade* grades;    // array of grades
  unsigned int grades_len;      // length of the array grades
} student;
```

You will need to implement three functions for managing students' grades. Make sure the functions are compatible with each other.

a) [7 points] Write a function that gets a name and an id of a student and returns a pointer to a new student with the given name and ID. The `grades` of the new student will be initialized to NULL, and `grades_len` set to 0 .

```
student* create_student(int id, char* name) {

  student* new_student = (student*) malloc(sizeof(student));
  if (new_student==NULL)
    return NULL;

  new_student->ID = id;
  new_student->name = name;
  new_student->grades = NULL;
  new_student->grades_len = 0;

  return new_student;




}
```

b) [8 points] Write a function that gets a pointer to a student and a course ID, and returns the grade of the student for this course. If the array of grades is NULL or the course is not in the array of grades, the function returns -1.

```
int find_grade(student* s, char* courseID) {
 // checking nulls
 if (s==NULL || courseID==NULL)
   return -1;

 // main loop
 for(int i=0; i<s->grades_len; i++)
   if (strcmp(s->grades[i].courseID, courseID)==0)
     return s->grades[i].grade;
 // if reached here, courseID not found
 return -1;
}
```

c) [10 points] Write a function that gets a pointer to a student and a course grade, and adds the grade to the student's array of grades.

If student->grades already contains a course with this courseID, the function does nothing.
If student->grades==NULL, you need to create an array of length 1 and add the new grade into the array.
If student->grades!=NULL, you need to increase the size of the array by one, and add new_grade into the new entry in the array of grades.

If a grade is added the function returns 1. Otherwise the function returns 0; this can be because the course is already there or malloc fails.

```
int add_grade(student* s, course_grade new_grade) {
  // checking nulls
  if (s==NULL || find_grade(s, new_grade.courseID)>-1)
    return 0;

  if (s->grades == NULL) { // if s->grades does not exist yet
    s->grades = (course_grade*) malloc(sizeof(course_grade));
    if (s->grades == NULL)
      return 0;
  }
  else   { // s->grades already exists
    s->grades = realloc(s->grades,(s->grades_len+1)*sizeof(course_grade));
    // we should handle realloc failure here...
  }
  s->grades[s->grades_len].grade=new_grade.grade; //copy  grade
  s->grades[s->grades_len].courseID=new_grade.courseID; //copy courseID
  s->grades_len=s->grades_len+1;  // update the length of the array
  return 1;
}
```

Problem 4 [25 points]

a) [5 points] Write a function that gets a string and reverses ***using a loop***.

```
void str_reverse_iter(char *str){

 int len = strlen(str);
 for (int i=0;i<len/2;i++) {
   char tmp = str[i];
   str[i] = str[len-1-i];
   str[len-i-1] = tmp;
 }









}
```

b) [10 points] Write a ***recursive*** function that gets a string and reverses it.
Loops are not allowed!

```
void str_reverse_rec(char *str){

 int len = strlen(str);
 if (len<=1)   // stopping condition
   return;
 else {
   char tmp = str[0]; // save str[0]
   str[0] = str[len-1]; // save the str[0] to be the last char
   str[len-1] = '\0'; // set the last char as null to reduce the length
   str_reverse_rec(str+1); // recursion on str[1...len-2]
   str[len-1] = tmp; // set the last char to be the original str[0]
 }



}
```

c) [10 points] Consider the following recursive function on non-negative integers.

```
foo(0) = 0
foo(1) = 1
foo(n) = foo(foo(i-1))+foo(i-1);  for n > 2.
```

Write a function that gets an integer `n` and computes `foo(n)`.
The function must work in O(n) time on the input n>0.

```
long foo(int n) {
```

ERROR IN THE QUESTION - CANCELLED

```
}
```