The University of Melbourne
School of Computing and Information Systems

# COMP10002
# Foundations of Algorithms
# Sample Exam 2 – Answers, Semester 1, 2018

Student ID:

**Total marks:** 60 **Length:** this exam has 10 pages

**Reading Time:** fifteen minutes **Writing Time:** two hours

**Identical Examination Papers:** none **Common Content Papers:** none

**Authorised Materials:**

- Writing materials, e.g., pens, pencils, are allowed.
- Books, calculators, and dictionaries are not allowed.

**Instructions to Invigilators:**

- Supply students with standard script book(s).
- The exam paper must be returned with all the written script book(s) to the subject coordinator.

**Instructions to Students:**

- Answer all questions. *All answers are to be written in the script book(s).*
- You may answer the questions in any order. However, you should write all of the answers that belong to the same question together.
- You are not required to write comments in any of your code fragments or functions except when you are explicitly asked to. If a question says "write a function", you may write relevant further functions if you believe that a decomposition of the problem is relevant.
- You may make use of library functions except when their use is explicitly prohibited. If you do make use of library functions, you must add suitable `#include` lines at the start of each corresponding answer.
- Constants should be `#define`'d prior to use, when appropriate.

**This paper may *not* be held by the Baillieu Library.**

1. Short answer questions [**15 marks in total**]

**1.1** [**2 marks**] Assume a 16-bit floating point number system where the most significant bit represents the sign; the following 3 bits represent an integer exponent of 2 with two's complement representation; and the rest of the bits represent the mantissa. For example, decimal number 0.375 is represented by 0 111 1100 0000 0000 in this system. Then for decimal number 0.625, what will it be represented by in this system?

*Answer:* 0 000 1010 0000 0000

**1.2** [**2 marks**] Name the two programs introduced in the subject for C debugging and profiling.

*Answer:* gdb, gprof

**1.3** [**2 marks**] Describe in no more than five English sentences the procedure of determining whether a given integer `x` is in a sorted array `A` of `n` integers using the divide and conquer strategy.

*Answer:*

1. Check the middle integer `m` in the array
2. If `m == x`, return 1
3. If `m > x`, recursively check (only) the first half
4. If `m < x`, recursively check (only) the second half
5. If `x` is not found in either half, return 0

**1.4** [**6 marks**] Suppose you are sorting an `int` array `array = {1, 3, 5, 2, 7, 6, 2, 4}` with the merge sort algorithm. Write out the array after each round of merging.

*Answer:*

1. `1, 3, 2, 5, 6, 7, 2, 4`
2. `1, 2, 3, 5, 2, 4, 6, 7`
3. `1, 2, 2, 3, 4, 5, 6, 7`

**1.5** [**3 marks**] Using the three-way partition strategy of the quick sort algorithm described in the lecture, what will the array `array = {1, 3, 5, 2, 7, 6, 2, 4}` look like after a partition using 4 as the pivot.

*Answer:* `1, 3, 2, 2, 4, 6, 7, 5`

**Programming and algorithm questions [45 marks in total]**

**2. [10 marks in total]**

**2.1** [**3 marks**] Write a function

`int is_perfect_number(int m)`

that returns 1 if `m` is a "perfect number" and 0 otherwise. Here, a perfect number is a positive number that equals to the sum of its factors (including the factor `1`, but excluding the number itself, and `1` is not a perfect number). For example, `6` is a perfect number because `6 = 1 + 2 + 3`.

*Answer:*

```
int is_perfect_number(int m) {
    int i, factor_sum = 0;

    if (m <= 1) {
        return 0;
    }

    for (i = 1; i < m; i++) {
        if (m % i == 0) {
            factor_sum += i;
        }
    }

    return factor_sum == m;
}
```

**2.2** [**2 marks**] Analyse the time complexity of the `is_perfect_number()` function.

*Answer:* The function takes `O(m)` time to run as it needs to loop from 1 to `m`.

**2.3** [**5 marks**] Write another function

`int count_perfect_number(int array[], int n)`

that calls the `is_perfect_number()` function and returns the number of perfect numbers in the `array` of `n (n > 0)` integers.

You need to add suitable `#include` lines if you use any library functions.

*Answer:*

```
int count_perfect_number(int array[], int n){
    int i, count = 0;

    for (i = 0; i < n; i++) {
        count += is_perfect_number(array[i]);
    }
    return count;
}
```

**3.** [**5 marks**]

Write a function

```
int str_to_int(char *str)
```

that converts a string `str` into the corresponding integer and returns the integer. For example, if `str =` "123", then the function should return `123`. You need to consider positive numbers (without the '`+`' sign), negative numbers, and zero.

You may assume that `str` is always valid (that is, can be converted into an integer). You may NOT make use of any functions in the `<string.h>` or `<stdlib.h>` libraries.

*Answer:*

```
#define POSITIVE 1
#define NEGATIVE -1
#define NEGATIVE_SIGN '-'
#define CHAR_ZERO '0'

int str_to_int(char *str) {
    int n = 0;
    int i = 0;
    int sign = POSITIVE;

    if (str[0] == NEGATIVE_SIGN) {
        sign = NEGATIVE;
        i = 1;
    }

    while (str[i]) {
        n = n * 10 + str[i] - CHAR_ZERO;
        i++;
    }

    return n * sign;
}
```

**4. [20 marks]**

In COMP10002, a student's record consists of a student ID represented by an integer, four marks (for the mid-semester exam, Assignment 1, Assignment 2, and the final exam) represented by four `double` typed numbers, and a grade represented by a character.

**4.1 [3 marks]** Write the definition of a `struct` type named `student_t` according to the description above.

*Answer:*

```
#define NUM_MARKS 4

typedef struct {
    int id;
    double marks[NUM_MARKS];
    char grade;
} student_t;
```

Assume that you are given a `student_t` typed array named `students` which contains the records of `n` students (`n > 0`). Each record has already got a student ID and the four marks, but the grade field does not have a value yet.

**4.2** [**7 marks**] Write a function

`void fill_grade(student_t students[], int n)`

that fills in the grade for each student according to the following rules:

If the student has failed the subject, then the grade should be ‘`F`’. Note that a student is considered failed if his/her total mark is less than `50`, or his/her total exam mark is less than `28`, or his/her total assignment mark is less than `12`. If the student has a total mark of `80` or above, then the grade should be ‘`H`’. All other students should be graded ‘`P`’.

You need to add suitable `#include` lines if you use any library functions.

*Answer:*

```
#define MID_EXAM 0
#define FINAL_EXAM 1
#define ASSMT_1 2
#define ASSMT_2 3
#define HONOUR 80
#define FAIL 50
#define EXAM_HURDLE 28
#define ASSMT_HURDLE 12

void fill_grade(student_t students[], int n) {
    int i;
    double exam_total, assmt_total, overall_total;

    for (i = 0; i < n; i++) {
        exam_total = students[i].marks[MID_EXAM] +
            students[i].marks[FINAL_EXAM];
        assmt_total = students[i].marks[ASSMT_1] +
            students[i].marks[ASSMT_2];
        overall_total = exam_total + assmt_total;

        if (exam_total < EXAM_HURDLE ||
            assmt_total < ASSMT_HURDLE ||
            overall_total < FAIL) {
            students[i].grade = ‘F’;
        } else if (overall_total >= HONOUR) {
            students[i].grade = ‘H’;
        } else {
            students[i].grade = ‘P’;
        }
    }
}
```

**4.3** [**4 marks**] Now write another function

`int student_id_cmp(void *student1, void *student2)`

that compares two students `student1` and `student2` by their IDs.

If the ID of `student1` is smaller than that of `student2`, then the function should return `1`. Otherwise the function should return `-1`.

You may assume that neither pointer is NULL.

*Answer:*

```
int student_id_cmp(void *student1, void *student2) {
    student_t *p1 = (student_t *)student1, *p2 = (student_t *)student2;

    if (p1->id < p2->id) {
        return 1;
    }
    return -1;
}
```

**4.4** [**6 marks**] You are given the follow type and function definitions:

```
typedef struct node node_t;
struct node {
    void *data;             /* ptr to stored structure */
    node_t *left;           /* left subtree of node */
    node_t *rght;           /* right subtree of node */
};
typedef struct {
    node_t *root;           /* root node of the tree */
    int (*cmp)(void*,void*); /* function pointer */
} tree_t;

tree_t *make_empty_tree(int func(void*,void*));
int is_empty_tree(tree_t *tree);
tree_t *insert_in_order(tree_t *tree, void *value);
void traverse_tree(tree_t *tree, void action(void*));
void free_tree(tree_t *tree);
```

Write a function

```
tree_t *insert_tree(student_t students[], int n)
```

that creates a binary search tree using `student_id_cmp` as the comparator function, inserts *a copy of* the student data from the array `students` into the tree, and returns the tree.

You may assume that `n > 0` and that the student IDs are all unique.

*Answer:*

```
#include <assert.h>
tree_t *insert_tree(student_t students[], int n) {
    tree_t *tree = make_empty_tree(student_id_cmp);
    student_t *new;
    int i = 0, j;

    while (i < n) {
        /* need to create a copy first before insertion */
        new = (student_t *)malloc(sizeof(student_t));
        assert(new);

        new->id = students[i].id;
        for (j = 0; j < NUM_MARKS; j++) {
            new->marks[j] = students[i].marks[j];
        }
        new->grade = students[i].grade;
        insert_in_order(tree, new);

        i++;
    }
    return tree;
}
```

**5. [5 marks]**

Write a recursive function

`int fibonacci(int n)`

that calculates and returns the Fibonacci number `F(n)`, where

`F(1) = F(2) = 1; F(n) = F(n-1) + F(n-2), n>2.`

You may assume `n > 0`.

If you use iteration rather than recursion to answer this question, the full mark of this question will reduce to 2 marks.

*Answer:*

```
int fibonacci(int n) {
    if (n == 1 || n == 2) {
        return 1;
    }
    return fibonacci(n-1) + fibonacci(n-2);
}

/* Iterative solution */
int fibonacci(int n) {
    int i = 3;
    int fibo, fibo1 = 1, fibo2 = 1;
    if (n == 1 || n == 2) {
        return 1;
    }

    while (i <= n) {
        fibo = fibo1 + fibo2;
        fibo1 = fibo2;
        fibo2 = fibo;
        i++;
    }

    return fibo;
}
```

**6. [5 marks]**

When evaluating mathematics expressions, it is important to make sure that the parentheses are correctly placed. Given a mathematics expression, we say that the parentheses are correctly placed if the number of opening parentheses and the number of closing parentheses are the same, and that within any prefix of the expression, the number of opening parentheses is not less than the number of closing parentheses. For example, "`1+(2*3)`" is a correct use of parentheses, while neither "`(1+2*3`" nor "`1)+(2*3`" is a correct use.

Write a function

```
int is_parentheses_correct(char *exp)
```

that checks a mathematics expression stored in a string `exp`, and returns if the parentheses are correctly used in `exp` (`1` for yes and `0` for no).

You may assume that `exp` is not `NULL`. You may NOT make use of any functions in the `<string.h>` library. You need to add suitable `#include` lines if you use any other library functions.

*Answer:*

```
int is_parentheses_correct(char *exp) {
    int i = 0, opening_count = 0;

    while (exp[i]) {
        if(exp[i] == '(') {
            opening_count++;
        }
        if(exp[i] == ')') {
            opening_count--;
            if (opening_count < 0) {
                return 0;
            }
        }
        i++;
    }

    return opening_count == 0;
}
```

**End of exam**