The University of Melbourne
School of Computing and Information Systems

COMP10002
Foundations of Algorithms
Sample Exam 1, Semester 1, 2018

Student ID:

Total marks: 60 **Length:** this exam has 5 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.

Short answer questions [15 marks in total]

1.1 [2 marks] In a 16-bit two's complement number representation for integers, what bit patterns represent the decimal numbers 123 and -123, respectively?

1.2 [2 marks] State two desired properties of numeric processing algorithms (among those listed in the lecture slides) that are different from the desired properties of symbolic processing algorithms.

1.3 [2 marks] State two facilities provided by the C preprocessor.

1.4 [6 marks] You are given an empty binary search tree `T` and an array of strings:

{"`apple`", "`banana`", "`pear`", "`watermelon`", "`cherry`", "`algorithms are fun`"}

Draw the tree `T` after each of the strings is inserted into it.

1.5 [3 marks] You are given a pattern string `str =` "`apple's app store`". Write the failure function values `F[0], F[1], ..., F[16]` corresponding to the pattern string.

Programming and algorithm questions [45 marks in total]

2. [5 marks]

Write a function

```
void reverse_array(int array[], int n)
```

that reverses the order of `n` integers in `array`.

For example, if `array = {1, 3, 8, 6, 2}` and `n = 5` is given to the function, then after calling the function, `array` should become `{2, 6, 8, 3, 1}`.

You may assume that `array` contains at least one integer. You may NOT define any new arrays in the `reverse_array()` function (that is, you must operate on `array` itself only).

3. [5 marks]

Write a function

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

that returns the number that appears for the most times in an array `array` of `n` integers. If there is a tie, return the smaller number.

For example, if `array = {1, 3, 2, 8, 3, 6, 2, 3}` and `n = 8`, then the function should return 3; if `array = {1, 2, 3, 2, 8, 3, 6, 2, 3}` and `n = 9`, then the function should return 2.

You may assume that `array` contains at least one integer.

4. [10 marks in total]

4.1 [7 marks] Write a function

```
void my_str_cat(char *dst, char *src)
```

that appends string `src` to the end of string `dst`.

For example, if `dst =` "`abc`" and `src =` "`def`", then the call `my_str_cat(dst, src)` will change `dst` to "`abcdef`".

You may assume that `dst` is not `NULL`, and that it has sufficient space to store any new character from `src`. You may NOT change `src` or make use of any functions in the `<string.h>`.

4.2 [3 marks] Analyse the time complexity of the `my_str_cat()` function.

5. [15 marks in total]

5.1 [5 marks] Write a function

```
int intersect(rectangle_t rect1, rectangle_t rect2)
```

that returns `1` if the two rectangles `rect1` and `rect2` interest each other, and `0` otherwise.

Here, each rectangle is represented by the lower and upper bounds in the x-dimension and the lower and upper bounds in the y-dimension, denoted by `lx, ux, ly, uy`, respectively. These bounds should be stored by `int` typed variables.

You need to first write out the definition of the `rectangle_t` type.

To check whether `rect1` and `rect2` intersect, your function `intersect()` needs to check whether the bounds of the two rectangles overlap with each other in both dimensions. Note that if the two rectangles only overlap at a vertex or an edge, they are still considered intersecting.

5.2 [5 marks] Now write another function

```
int rect_cmp(void *rect1, void *rect2)
```

that compares two rectangles `rect1` and `rect2` by their size.

If `rect1` is smaller than `rect2` in size, then `rect_cmp(rect1, rect2)` should return `1`. If `rect1` is larger than `rect2` in size, then `rect_cmp(rect1, rect2)` should return `-1`. If the two rectangles have the same size, then `rect_cmp(rect1, rect2)` should return `0`.

You may assume that neither pointer is NULL.

5.3 [**5 marks**] You are given the following type and function definitions:

```
typedef struct node node_t;
typedef rectangle_t data_t;

struct node {
    data_t data;
    node_t *next;
};

typedef struct {
    node_t *head;
    node_t *foot;
} list_t;

list_t *make_empty_list(void);
void insert_at_head(list_t *list, data_t value);
void insert_at_foot(list_t *list, data_t value);
```

Write a `main()` function that first creates an empty list, then repeatedly prompts a user to input the four bounds of a rectangle and inserts the rectangle to the end of the list until no more input has been entered by the user. The `main()` function should then print out the rectangles in the list from the `head` to the `foot`, one at a line, and then frees the memory allocated for the list.

6. [**5 marks**]

Write a recursive function

```
int is_palindrome(char *str, int n)
```

that returns `1` if `str` is a palindrome, that is, reads exactly the same forwards as well as backwards. If `str` is not a palindrome, then the function should return `0`. Here, `n` is the length of the string `str`.

For example, if `str =` "`rats live on no evil star`", then `is_palindrome(str, 25)` should return `1`. If `str =` "`abab`", then `is_palindrome(str, 4)` should return `0`.

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

7. [5 marks]

Write a function

```
int randomised_subset_sum(int items[], int n, int k)
```

that uses a randomised strategy to solve the subset sum problem with the set of `n` items represented by the items array, and the sum to achieve represented by `k`.

This randomised strategy repeats the following steps for 10,000 iterations. At each iteration, it randomly chooses an integer `num (0 < num <= n)`. Then it randomly chooses `num` items from the items array. If these `num` items add to `k`, then `randomised_subset_sum()` returns `1`. Otherwise, it starts the next iteration. When 10,000 iterations are completed, `randomised_subset_sum()` returns `0`.

When choosing the `num` items randomly, you need to find a way to guarantee that no item is chosen twice from the items array.

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

End of exam