UNIVERSITY OF BRISTOL

May/June 2022 Examination Period

FACULTY OF ENGINEERING

First Year Examination for the Degrees of
Bachelor of Engineering
Master of Engineering
Bachelor of Science

COMS-10017
Object-Oriented Programming and Algorithms I

TIME ALLOWED:
2 Hours

This paper contains *three* questions.
All answers will be used for assessment.
The maximum for this paper is *100 marks*.

PLEASE WRITE YOUR 7 DIGIT STUDENT NUMBER (NOT CANDIDATE NUMBER) ON THE ANSWER BOOKLET. YOUR STUDENT NUMBER CAN BE FOUND ON YOUR UCARD.

Other Instructions:

1. **Calculators must have the Faculty of Engineering Seal of Approval.**

TURN OVER ONLY WHEN TOLD TO START WRITING

Important Information: Throughout this exam paper log() denotes the binary logarithm, i.e, $\log(n) = \log_2(n)$. We also write $\log\log n$ as an abbreviation for $\log(\log(n))$, and $\log^c n$ as an abbreviation for $(\log n)^c$.

As in the lectures, arrays start at index 0. For example, an array A of length n consists of the elements $A[0], A[1], \ldots, A[n-1]$.

Q1. This question is about Big-O notation and loop invariants.

(a) Which of the following functions is in $O(n)$?

- ☐ $\frac{1}{2}n + 5$
- ☐ $\sqrt{n}\log^2 n$
- ☐ $n^2/\log n$

[3 marks]

(b) Which of the following statements is true?

- ☐ $7 \in O(1)$
- ☐ $3n^2\log n \in O(n^2)$
- ☐ $n + \log^2 n \in O(n)$

[3 marks]

(c) Which of the following statements is true?

- ☐ $4\log(2n) \in \Theta(\log_{10} n)$
- ☐ $n! \in O(2^n)$
- ☐ $\sum_{i=1}^{\sqrt{n}\log n} i \in O(n\log n)$

[3 marks]

(d) Which of the following statements is true?

- ☐ There exist functions $f, g \in \Theta(\sqrt{n})$ such that $f + g = \Omega(n)$?
- ☐ There exist functions $f, g \in \Theta(n)$ such that $f - g = \Theta(\log n)$?
- ☐ There exist functions f, g such that $f \in O(g)$ and $2^f \notin O(2^g)$?

[3 marks]

(e) Let $f, g : \mathbb{N} \to \mathbb{N}$ be functions. For each of the following statements, mark whether the statement, potentially together with an application of the racetrack principle, implies that $f(n) \in O(g(n))$.

- ☐ $f(4) \leq \frac{1}{2}g(4)$ and $g'(n) \geq f'(n)$, for every $n \leq 100$
- ☐ $f(10) \leq 10 \cdot g(10)$ and $g'(n) \geq f'(n)$, for every $n \geq 100$
- ☐ f, g are increasing functions, $f(50) \leq g(25)$, and $g'(n) \geq f'(n)$, for every $n \geq 2$
- ☐ f, g are increasing functions, $f(17) \leq g(20)$, and $g'(n) \geq f'(n)$, for every $n \geq 15$

[8 marks]

(f) Order the following sets so that each is a subset of the one that comes after it:

(1) $O(\log(n)^{\log\log n})$ (2) $O(\sqrt{n}^{3/2})$ (3) $O(n!)$ (4) $O(2^{n \log\log n})$
(5) $O(\sum_{i=1}^{n} i)$ (6) $O(\log^2 n)$ (7) $O(2^{\sqrt{\log\log n}})$ (8) $O(\frac{n}{e^2})$

[6 marks]

(g) Consider the algorithm Strange Algorithm listed in Algorithm 1:

Algorithm 1 Strange Algorithm

```
Require: Integer n
1: S ← 0
2: for i ← 1 ... n do
3:     if i is odd then
4:         S ← S − i
5:     else
6:         S ← S + i
7:     end if
8: end for
9: return S
```

Which of the following loop-invariants is correct?

At the beginning of iteration i (i.e., after i is updated in Line 2 and before the code in Line 3 is executed) the following property holds:

- ☐ $S = (-1)^{i+3}((i-1)/2)$ if i is odd and $S = (-1)^{i+1} i/2$ if i is even
- ☐ $S = -(i+1)/2$ if i is odd and $S = i/2$ if i is even
- ☐ $S = i/2 - 0.5$ if i is odd and $S = -i/2$ if i is even
- ☐ $S = (-1)^{i}(i/2)$ if i is odd and $S = (-1)^{i}(i-1)/2$ if i is even

[5 marks]

(h) Consider the algorithm Very Strange Algorithm listed in Algorithm 2:

Algorithm 2 Very Strange Algorithm

```
Require: Integer n
S ← 0
for i ← 1 ... n do
    S ← S + 2^(n−i) − 2^(i−1)
end for
return S
```

What are the correct values for **X**, **Y**, and **Z** in order to obtain a correct loop invariant:

At the beginning of iteration i (i.e., after i is updated in Line 2 and before the code in Line 3 is executed) the following property holds:

$$S = \mathbf{X} - 2^{\mathbf{Y}} + 2^n(1 - \frac{1}{2^{\mathbf{Z}}}) .$$

Hint: Recall that $\sum_{j=0}^{i} 2^j = 2^{i+1} - 1$.

[6 marks]

Q2. This question is about sorting.

(a) Mark each of the following statements with either true or false:

- ☐ Quicksort is a dynamic programming algorithm.
- ☐ Mergesort is a divide-and-conquer algorithm.
- ☐ Radixsort is a divide-and-conquer algorithm.
- ☐ Heapsort requires a tree data structure in order to store the tree representation of the input array.
- ☐ Heapsort has a runtime of $O(n)$ on an input array A of length n with $A[i] = 1$, for every i.
- ☐ No sorting algorithm can sort an array of length n consisting of distinct integers taken from the set $\{1, 2, \dots, n^{10}\}$ in time $O(n)$.
- ☐ Radixsort uses a stable sorting algorithm as a subroutine.
- ☐ The best-case, average-case, and worst-case runtimes of Mergesort are identical (in big-Θ-notation).
- ☐ Countingsort counts how often each element appears in the input array.

[9 marks]

(b) For each of the following inputs to Insertionsort, state the runtime of the algorithm using $\Theta(.)$-notation (no justification needed; the objective is to sort in increasing order):

1. Integer array A_1 of length n with $A_1[i] = i$ for every $i \leq n/2$, and $A_1[i] = n - i$ for every $i > n/2$. For example, if $n = 6$, then $A_1 = 0\ 1\ 2\ 3\ 2\ 1$.
2. Integer array A_2 of length n with $A_2[i] = 4i + 6$ for every $0 \leq i \leq n - 1$.
3. Integer array A_3 of length n with $A_3[i] = 1$ for every $i \in \{1 \cdot n^{1/3}, 2 \cdot n^{1/3}, 3 \cdot n^{1/3}, \dots\}$ and $A_3[i] = i$ otherwise. We assume that $n^{1/3}$ is an integer here.

[6 marks]

(c) Consider the following integer array A of length 11:

$$A = 7 \quad 5 \quad 11 \quad 3 \quad 8 \quad 12 \quad 2 \quad 1 \quad 5 \quad 6 \quad 4$$

Heapsort interprets A as a binary tree. As in the lecture, we assume that the employed heap is a max-Heap, i.e., the maximum is at the root. Suppose that we run the BUILD-HEAP() operation on A.

1. How many node exchanges take place overall?
2. Indicate the values of the nodes that were exchanged in the last exchange.

[7 marks]

(d) We use Radixsort to sort the following array A of integers in increasing order:

$$A[0] = 2374, A[1] = 1809, A[2] = 7888, A[3] = 9124, A[4] = 6844, \ .$$

Let $A_0 = A$ and denote by A_i the array A at the end of iteration i of the main loop of Radixsort.

In how many iterations of the main loop of Radixsort does the relative order of the numbers 1809 and 9124 in the input array change, i.e., how many $i \geq 1$ are there such that the relative order of the numbers 1809 and 9124 in A_i and A_{i-i} is different?

[5 marks]

(e) Recall that the PARTITION function in Quicksort reorders the input array of length n so that all elements smaller than the pivot are to the left of the pivot and all elements larger than the pivot are to the right of the pivot. Denote by n_1 the number of elements to the left of the pivot and by n_2 the number of elements to the right of the pivot. Then, we say that a selected pivot is *good* if $\min\{n_1, n_2\} \geq \frac{1}{3}n$.

Consider now the recursion tree of an execution of Quicksort. We say that a node in the recursion tree is good if the recursive call to Quicksort that corresponds to this node chose a good pivot.

Suppose now that, on any root-to-leaf path in the recursion tree, among every t consecutive nodes there is at least one good node. What is then the worst-case runtime of Quicksort if:

1. $t = 5$
2. $t = \log n$

[4 marks]

Q3. This question concerns algorithmic design principles and recurrences.

(a) Consider the pole cutting problem from the lecture with the following price function p:

length i	1	2	3	4	5	6	7	8	9
price $p(i)$	1	5	7	8	8	11	12	16	19

Let $r(i)$ be the maximum revenue achievable for a pole of length i. What are the values $r(2)$,$r(3)$,$r(4)$,$r(5)$,$r(6)$,$r(7)$,$r(8)$,$r(9)$?

[8 marks]

(b) Determine the runtimes of Algorithms 3, 4, and 5 using Big “Theta” notation.

(cont.)

Algorithm 3

Require: Int $n \geq 1$

```
x ← 0
for i = 1 ... n do
    for j = 1 ... ⌈i/2⌉ do
        x ← x + i · j
    end for
end for
return x
```

Algorithm 4

Require: Int $n \geq 1$

```
x ← 0
i ← 1
while i ≤ n do
    i ← 2 · i
    for j = i ... 4 · i do
        x ← x + i · j
    end for
end while
return x
```

Algorithm 5

Require: Int $n \geq 1$

```
x ← 0
i ← 1
while i ≤ n do
    i ← log(n) · i
    for j = 1 ... n do
        x ← x + i · j
    end for
end while
return x
```

[9 marks]

(c) Let **A** be a recursive algorithm that takes a real number n as its input. Algorithm **A** recursively invokes itself k times on input $n/2$, for some integer k, if $n > 1$, and does not invoke itself recursively if $n \leq 1$. Without the recursive calls, the runtime of algorithm **A** on real number n is $\Theta(n)$.

1. Suppose that $k = 2$. How often does **A** invoke itself (in Θ-notation) and what is the total runtime of **A** (in Θ-notation)?
2. Suppose now that $k = 1$. How often does **A** invoke itself (in Θ-notation) and what is the total runtime of **A** (in Θ-notation)?
3. Suppose now that $k = \log n$. How often does **A** invoke itself (in Θ-notation) and what is the total runtime of **A** (in Θ-notation)?

Hint: Draw a recursion tree.

[9 marks]

(d) Let $a, b \geq 1$ be integers. Consider the following recurrence:

$$\begin{aligned} T(n) &= a \cdot T(n-1) + b \cdot T(n-2) \text{ , for every } n \geq 3 \\ T(1) = T(2) &= 1 \text{ .} \end{aligned}$$

Which of the following statements is correct?

☐ $T(n) = \Theta\left((\frac{a}{2} + \sqrt{a^2 + 4b})^n\right)$

☐ $T(n) = \Theta\left((a + \sqrt{a^2 + 4b})^n\right)$

☐ $T(n) = \Theta\left((a + \sqrt{\frac{a^2}{4} + b})^n\right)$

☐ $T(n) = \Theta\left((a + \frac{\sqrt{\frac{a^2}{4} + b}}{2})^n\right)$

☐ $T(n) = \Theta\left((\frac{a}{2} + \sqrt{\frac{a^2}{4} + b})^n\right)$

☐ $T(n) = \Theta\left((\frac{a}{2} + \frac{\sqrt{\frac{a^2}{4} + b}}{2})^n\right)$

(cont.)

☐ None of the above

Hint: Use the substitution method.

[6 marks]

END OF PAPER