

Computer science
Standard level
Paper 2

Mock examination — Set 1

1 hour 15 minutes

Instructions to candidates

- Do not open this examination paper until instructed to do so.
- Answer all of the questions from **one** of the options.
- The maximum mark for this examination paper is **[50 marks]**.

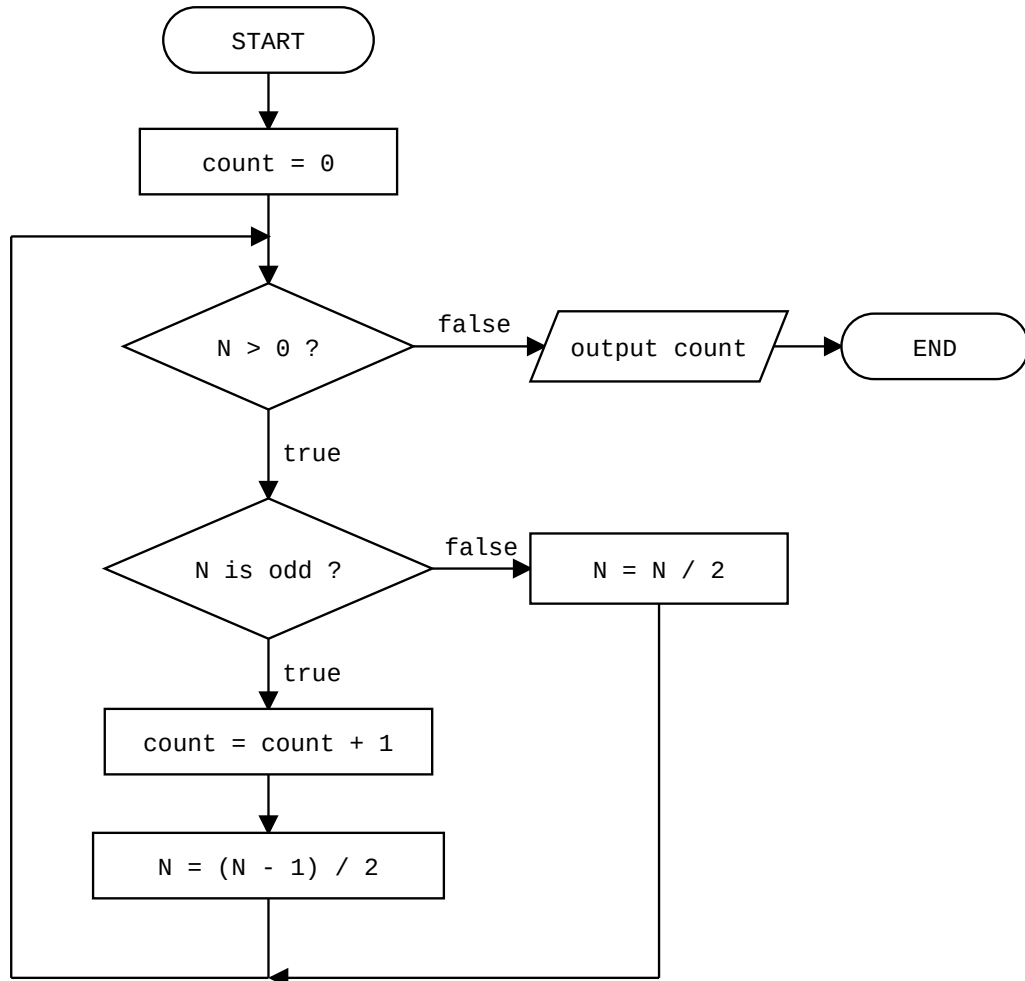
Option	Questions
Option A — Java	1 – 3
Option B — Python	4 – 6

Option A — Java

Answer **all** questions. All code that forms part of your responses must be written in Java.

- The flowchart in **Figure 1** represents an algorithm that processes a value, N , which is supplied when the algorithm starts.

Figure 1: Flowchart for an algorithm that processes N



- State **one** requirement for N that should be included in the input specification for this algorithm. [1]

- Copy and complete the trace table for this flowchart, where $N = 13$. [3]

N	$N > 0 ?$	N is odd ?	count	output
13			0	

The efficiency of an algorithm can be expressed using Big O notation.

- Describe the time complexity of the algorithm in **Figure 1**, using Big O notation. [2]

(Option A continues on the following page)

(Option A, question 1 continued)

Table 1 shows a list of values that has been sorted into ascending order.

Table 1: A sorted list of values

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Value	5	11	16	23	29	34	42	47	53	58	64	70	77	81	96

- (d) Describe the steps that a binary search would follow to find the value 29 in this list. [4]

Queues are used in computer science to process items in the order in which they arrive.

Table 2 shows the fundamental operations that operate on a queue.

Table 2: Fundamental operations that operate on a queue

Operation	Description
enqueue(x)	Adds the element x to the back of the queue.
dequeue()	Removes the element at the front of the queue and returns it.
front()	Returns the element at the front of the queue without removing it from the queue.
isEmpty()	Returns true if the queue is empty, else it returns false.

The following operations are carried out, in order, on an empty queue:

enqueue(7)
enqueue(3)
enqueue(12)
dequeue()
front()
enqueue(5)
dequeue()
enqueue(9)

- (e) Sketch the queue after these operations have been carried out.

You should clearly indicate the front and the back of the queue. [2]

A different queue contains whole numbers. Some of these numbers are negative.

- (f) Using only the queue operations shown in **Table 2**, describe the steps that will remove all the negative numbers from the queue, so that the remaining numbers stay in their original order. [3]

(Option A continues on the following page)

(Option A continued)

2. A ferry company sells tickets for crossings between island ports. Each ticket shows a sailing code made of the departure port, the arrival port and the departure time, separated by hyphens. For example, the sailing code `ARD-BRE-0915` is for the 09:15 crossing from the port `ARD` to the port `BRE`. Each port code has exactly three letters and each departure time has exactly four digits.

Each ferry can carry a maximum of 150 passengers. For each sailing, the booking system stores the names of the passengers.

- (a) Outline **one** advantage of using a static data structure, rather than a dynamic data structure, to store the names of the passengers for a sailing. [2]

The numbers of passengers on six sailings are stored in the array `passengers`:

```
int[] passengers = {48, 112, 35, 90, 61, 27};
```

The array is sorted into ascending order using a bubble sort. In each pass, adjacent elements are compared, starting at the beginning of the array.

- (b) State the contents of `passengers` after each of the **first two** passes of the bubble sort. [2]

A ticket clerk types the number of tickets that a customer wants to buy. The method `readTickets()` converts the text typed by the clerk into an integer:

```
public static int readTickets(String text)
{
    int tickets = 0;
    try
    {
        tickets = Integer.parseInt(text);
    }
    catch (NumberFormatException e)
    {
        System.out.println ("Enter a whole number");
    }
    finally
    {
        System.out.println ("Input checked");
    }
    return tickets;
}
```

- (c) Describe what happens when `readTickets("3x")` is called. [2]

(Option A continues on the following page)

(Option A, question 2 continued)

The fares paid for five tickets on one sailing are stored in the array `fares`. The following code is intended to output the lowest of these fares:

```
double[] fares = {14.50, 9.75, 22.00, 9.75, 18.25};
double lowest = 0;
for (int i = 0; i < 5; i++)
{
    if (fares[i] < lowest)
        lowest = fares[i];
}
System.out.println ("Lowest fare: " + lowest);
```

- (d) Identify the logic error in this code and outline how it should be corrected. [2]

The method `departureTime()` is passed a sailing code and returns its departure time in the format hours:minutes. For example, `departureTime("ARD-BRE-0915")` returns "09:15".

- (e) Construct the method `departureTime()`. [3]

At the end of each day, the sailing code of every ticket sold that day is written to the text file `tickets.txt`, one sailing code per line. The first three lines of the file are shown:

```
ARD-BRE-0915
CAL-ARD-1030
ARD-CAL-1400
...
```

The method `countFrom()` is passed a three-letter port code. It reads `tickets.txt` line by line until the end of the file is reached and then outputs the number of tickets sold for sailings that depart from that port. For example, `countFrom("ARD")` outputs the number of tickets sold for sailings that depart from ARD. The file is currently closed.

- (f) Construct the method `countFrom()`. [4]

(Option A continues on the following page)

(Option A continued)

3. A music festival uses object-oriented programming (OOP) to manage the acts that perform on its stages.

During the festival, members of the audience use an app to rate each performance with a whole number from 1 to 5. The ratings for each act are stored in the `Performer` object for that act. Each act receives at most 2000 ratings.

The `Performer` class is partially shown:

```
public class Performer
{
    private static int maxRating = 5; // highest rating allowed
    private String name;               // name of the act
    private String stage;              // stage on which the act performs
    private int[] ratings;             // ratings given by the audience
    private int numRatings;            // number of ratings stored so far

    public Performer(...)
    {
        // code missing
    }

    public String getName()
    {
        return this.name;
    }

    public String getStage()
    {
        return this.stage;
    }

    public void addRating(int r) // stores r if it is a valid rating
    {
        if (r >= 1 && r <= maxRating)
        {
            this.ratings[this.numRatings] = r;
            this.numRatings = this.numRatings + 1;
        }
    }

    public double averageRating() // returns the mean of the ratings
    {
        // code missing
    }
}
```

- (a) Construct a UML class diagram for the `Performer` class. [3]

The attribute `maxRating` is declared as static.

- (b) Distinguish between the attribute `maxRating` and the non-static attribute `numRatings`. [2]

(Option A continues on the following page)

(Option A, question 3 continued)

The attributes `maxRating`, `ratings` and `numRatings` are declared as `private`.

- (c) Explain how information hiding helps to ensure that a `Performer` object stores only valid ratings. [2]

When a `Performer` object is created, the constructor is passed the name of the act and the stage on which it performs. The constructor must also prepare the object to store up to 2000 ratings, with no ratings stored yet. For example:

```
Performer act1 = new Performer("The Lanterns", "Lakeside");
```

- (d) Construct the constructor for the `Performer` class. [3]

The method `averageRating()` returns the mean of the ratings stored for an act. If no ratings have been stored, it returns 0.

- (e) Construct the method `averageRating()`.

The `Arrays.stream()` and `IntStream` methods must not be used. [5]

The festival has 40 acts. An array, `lineup`, of size 40 holds the `Performer` objects, one for each act.

The driver class includes the method `showPerformer()`. This method is passed `lineup` and the name of an act. It searches `lineup` for that act and:

- outputs the stage and the average rating of the act if the act is found
- outputs `Not on the line-up` if the act is not found.

- (f) Construct the method `showPerformer()`.

You may use any of the methods in the `Performer` class. Assume that no two acts have the same name. [5]

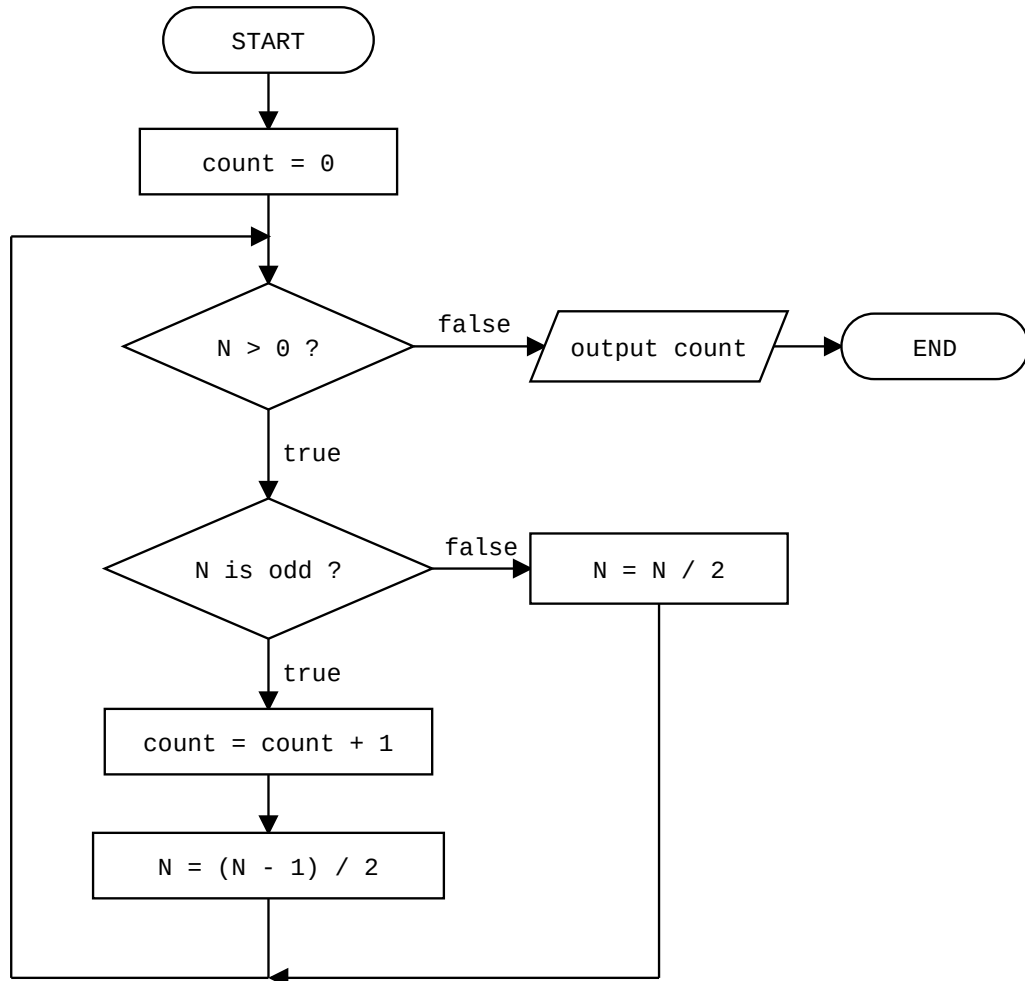
End of Option A

Option B — Python

Answer **all** questions. All code that forms part of your responses must be written in Python.

4. The flowchart in **Figure 2** represents an algorithm that processes a value, N , which is supplied when the algorithm starts.

Figure 2: Flowchart for an algorithm that processes N



- (a) State **one** requirement for N that should be included in the input specification for this algorithm. [1]

- (b) Copy and complete the trace table for this flowchart, where $N = 13$. [3]

N	$N > 0 ?$	N is odd ?	count	output
13			0	

The efficiency of an algorithm can be expressed using Big O notation.

- (c) Describe the time complexity of the algorithm in **Figure 2**, using Big O notation. [2]

(Option B continues on the following page)

(Option B, question 4 continued)

Table 3 shows a list of values that has been sorted into ascending order.

Table 3: A sorted list of values

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Value	5	11	16	23	29	34	42	47	53	58	64	70	77	81	96

- (d) Describe the steps that a binary search would follow to find the value 29 in this list. [4]

Queues are used in computer science to process items in the order in which they arrive.

Table 4 shows the fundamental operations that operate on a queue.

Table 4: Fundamental operations that operate on a queue

Operation	Description
enqueue(x)	Adds the element x to the back of the queue.
dequeue()	Removes the element at the front of the queue and returns it.
front()	Returns the element at the front of the queue without removing it from the queue.
isEmpty()	Returns true if the queue is empty, else it returns false.

The following operations are carried out, in order, on an empty queue:

enqueue(7)
enqueue(3)
enqueue(12)
dequeue()
front()
enqueue(5)
dequeue()
enqueue(9)

- (e) Sketch the queue after these operations have been carried out.

You should clearly indicate the front and the back of the queue. [2]

A different queue contains whole numbers. Some of these numbers are negative.

- (f) Using only the queue operations shown in **Table 4**, describe the steps that will remove all the negative numbers from the queue, so that the remaining numbers stay in their original order. [3]

(Option B continues on the following page)

(Option B continued)

5. A ferry company sells tickets for crossings between island ports. Each ticket shows a sailing code made of the departure port, the arrival port and the departure time, separated by hyphens. For example, the sailing code `ARD-BRE-0915` is for the 09:15 crossing from the port `ARD` to the port `BRE`. Each port code has exactly three letters and each departure time has exactly four digits.

Each ferry can carry a maximum of 150 passengers. For each sailing, the booking system stores the names of the passengers.

- (a) Outline **one** advantage of using a static data structure, rather than a dynamic data structure, to store the names of the passengers for a sailing. [2]

The numbers of passengers on six sailings are stored in the list `passengers`:

```
passengers = [48, 112, 35, 90, 61, 27]
```

The list is sorted into ascending order using a bubble sort. In each pass, adjacent elements are compared, starting at the beginning of the list.

- (b) State the contents of `passengers` after each of the **first two** passes of the bubble sort. [2]

A ticket clerk types the number of tickets that a customer wants to buy. The function `read_tickets()` converts the text typed by the clerk into an integer:

```
def read_tickets(text):
    tickets = 0
    try:
        tickets = int(text)
    except ValueError:
        print ("Enter a whole number")
    finally:
        print ("Input checked")
    return tickets
```

- (c) Describe what happens when `read_tickets("3x")` is called. [2]

The fares paid for five tickets on one sailing are stored in the list `fares`. The following code is intended to output the lowest of these fares:

```
fares = [14.50, 9.75, 22.00, 9.75, 18.25]
lowest = 0
for i in range (0, 5):
    if fares[i] < lowest:
        lowest = fares[i]
print ("Lowest fare: ", lowest)
```

- (d) Identify the logic error in this code and outline how it should be corrected. [2]

The function `departure_time()` is passed a sailing code and returns its departure time in the format `hours:minutes`. For example, `departure_time("ARD-BRE-0915")` returns `"09:15"`.

- (e) Construct the function `departure_time()`. [3]

(Option B continues on the following page)

(Option B, question 5 continued)

At the end of each day, the sailing code of every ticket sold that day is written to the text file `tickets.txt`, one sailing code per line. The first three lines of the file are shown:

```
ARD-BRE-0915
CAL-ARD-1030
ARD-CAL-1400
...
```

The function `count_from()` is passed a three-letter port code. It reads `tickets.txt` line by line until the end of the file is reached and then outputs the number of tickets sold for sailings that depart from that port. For example, `count_from("ARD")` outputs the number of tickets sold for sailings that depart from ARD. The file is currently closed.

- (f) Construct the function `count_from()`. [4]

(Option B continues on the following page)

(Option B continued)

6. A music festival uses object-oriented programming (OOP) to manage the acts that perform on its stages.

During the festival, members of the audience use an app to rate each performance with a whole number from 1 to 5. The ratings for each act are stored in the `Performer` object for that act. Each act receives at most 2000 ratings.

The instance attributes of each `Performer` object are `self._name` (the name of the act), `self._stage` (the stage on which the act performs), `self._ratings` (a list of the ratings given by the audience) and `self._num_ratings` (the number of ratings stored so far).

The `Performer` class is partially shown:

```
class Performer():

    _max_rating = 5                # highest rating allowed

    def __init__(...):
        # code missing

    def get_name(self):
        return self._name

    def get_stage(self):
        return self._stage

    def add_rating(self, r):        # stores r if it is a valid rating
        if r >= 1 and r <= Performer._max_rating:
            self._ratings.append(r)
            self._num_ratings = self._num_ratings + 1

    def average_rating(self):       # returns the mean of the ratings
        # code missing
```

- (a) Construct a UML class diagram for the `Performer` class. [3]

The attribute `_max_rating` is a class (static) attribute.

- (b) Distinguish between the attribute `_max_rating` and the instance (non-static) attribute `self._num_ratings`. [2]

The underscore character at the start of `_max_rating`, `_ratings` and `_num_ratings` identifies each of these variables as being a private attribute of the class.

- (c) Explain how information hiding helps to ensure that a `Performer` object stores only valid ratings. [2]

(Option B continues on the following page)

(Option B, question 6 continued)

When a `Performer` object is created, the constructor is passed the name of the act and the stage on which it performs. The constructor must also prepare the object to store the ratings, with no ratings stored yet. For example:

```
act1 = Performer("The Lanterns", "Lakeside")
```

- (d) Construct the constructor, `__init__()`, for the `Performer` class. [3]

The method `average_rating()` returns the mean of the ratings stored for an act. If no ratings have been stored, it returns 0.

- (e) Construct the method `average_rating()`.

The `sum()` function and the `statistics` module must not be used. [5]

The festival has 40 acts. A list, `lineup`, holds the 40 `Performer` objects, one for each act.

The driver code includes the function `show_performer()`. This function is passed `lineup` and the name of an act. It searches `lineup` for that act and:

- outputs the stage and the average rating of the act if the act is found
- outputs `Not on the line-up` if the act is not found.

- (f) Construct the function `show_performer()`.

You may use any of the methods in the `Performer` class. Assume that no two acts have the same name. [5]

End of Option B
