Test PrP Regular exam - 2023-2024

Maastricht University

Test ID: 108587
Folder: /Preview
Version: 1.4
Randomised: No
Last modified: Monday, 20 november 2023 09:32:00
Number of questons: 12
Question order: Fixed
Display questions once: No
Tools: None
Test time: 120 minutes
Maximum score: 100 pt.
Chance score: 3.20 pt. / 3%
In test set with: -

Question 1 – Theory–Java – 149628.4.1

Java is described by Sun Microsystems, the initial developers of Java, as WORA (Write Once Run Anywhere). However, to be able to run Java code on different machines, you need to use a "tool" that allows the code written once to be executed (run) on any machine. Which is this tool? What is the process one must follow to run Java code after writing your .java file in several types of machines (e.g. same code to be executed in Windows and Linux devices)? Explain this process in your own words. Do not use more than 50 words.

Grading instruction

Explanation (Number of points: 6)

Full points: The student provided an accurate description of the compilation-interpretation steps. The JVM is mentioned. The idea and differences between compiler and interpreter are clear.

Question 2 – Theory–For loop – 149672.3.0

What is the value of the variable `b` after the execution of the following statements?

```
int a=10, b=0;
for (int i=a;i>0;i=i-2) {
    b=b+i;
}
```

30

Question 3 − Theory−Find errors − 149632.3.2

The piece of code included below has some issues. There are several parts that if we try to compile (or execute), would throw a compilation (or runtime) error. Mark in the code below the parts of the code that will generate a problem. Be as precise as possible when highlighting the error (i.e. do not mark the line but the part of the line where the problem is located).

Note: There are four (4) errors and you must use all the markers to complete the exercise.

```
int a, b=-10;

System.out.println("Initial value of a: "+a);
for(i = 0; i>b; i++) {
    a = b+i
    if(a == 10)
        System.out.println("a reached the value 10.");
}

System.out.println("Final value of a: "a);
```

Question 4 − Theory−Analyse code − 149629.5.1

We have implemented the following method:

```
public static int outputMethod(int a, int b) {
    int result = 0;

    for(int i=a; i<b; i++) {
        result = b/a;
    }
    return result;
}
```

Unfortunately, some combinations of values for the input parameters `a` and `b` will provoke a compilation and/or runtime error. Select the option (only one) below that will let to an error in all the cases.

A `a = 0`

B `a < b`

C `a = 0` AND `a > b`

D `a = 0` AND `b > 0`

E `a = 0` OR `a < b`

Question 5 − Theory−Fill in array − 149626.1.2

Fill in the array values that would be stored after the code below executes in the table below.

```
int[] data = new int[8];
data[0] = 3;
data[7] = -18;
data[4] = 5;
data[1] = data[0];
int x = data[4];
data[4] = 6;
data[x] = data[0] * data[1];
```

3	3	0	0	6	9	0	-18

[Alphanumeric]	[Alphanumeric]	[Alphanumeric]	[Alphanumeric]	[Alphanumeric]
3	**3**	**0**	**0**	**6**
[Alphanumeric]	[Alphanumeric]	[Alphanumeric]		
9	**0**	**-18**		

Question 6 − Theory−Shrinking an array − 149687.2.1

We want to implement a piece of code that "shrinks" a given array. To that end, it creates a smaller version of a given 1D array (named as `initialArray`). The new array (named as `finalArray`) should contain all the elements in `initialArray` except for the last element. What is the purpose of line 12?
Note: The information printed in the last line will be `[2, 8, 4]` since, after this piece of code, `initialArray` contains one less element than it did at the beginning.

```
1  import java.util.Arrays;
2
3  public class MyArrayCopy {
4
5      public static void main(String[] args) {
6          int[] initialArray = {2,8,4,1};
7
8          int[] finalArray = new int[initialArray.length-1];
9          for(int i=0; i<finalArray.length; i++) {
10             finalArray[i] = initialArray[i];
11         }
12         initialArray = finalArray;
13
14         System.out.println(Arrays.toString(initialArray));
15     }
16 }
```

A The content of `finalArray` (values for all the elements) is copied into `initialArray`.

B Changing the reference of `initialArray` so now, it points to the new (reduced) array.

C The content of `finalArray` (values for all the elements) is copied into `initialArray except for the last element, which is ignored`.

Question 7 − Coding−Assignment − EXPLANATION − 149860.2.1

In this exercise, you must implement an algorithmic approach to what we call "cool" numbers. Given a (non-zero) positive integer `n`, a number `x` is considered "cool" if `n` is divisible by each of the number's digits (of `x`) without remainder.
Some examples follow:

- If n=16, then 2418 is a "cool" number because 16/2=8, 16/4=4, 16/1=16, 16/8=2 (none of these divisions leaves a remainder).
- If n=16, then 489 is not a "cool" number because 16/9=1 and leaves a remainder of 7. Note that the other digits are divisible with no remainder, however, recall that we want all of them to be divisible.
- If n=5, then 10000 is not a "cool" number since division by 0 is not possible -you need to take care of that in your program.

Write IN YOUR OWN WORDS, how you would solve the problem. Do not use pseudocode but a high-level explanation of how you plan to solve the problem.
IMPORTANT: This part is only evaluated if you also provide a "good" attempt to implement it in the next question (coding part).

Grading instruction

Criterion 1 (Number of points: 10)

Grading criteria:
For full grade, we expect that the explanation includes:

- Filtering if x is zero.
- Filtering zeros within x (digits).
- How to get the digits (e.g. dividing by 10 consecutively) is clearly explained.
- The use of the reminder (ideally mentioning %) is clearly explained.
- When to return true AND false is clear (e.g. you do not return true once you find one reminder = 0, etc.).

We use the following legend to indicate the missing parts:
x: not met (either missing or wrong)
~: partially met (e.g. the idea is correct but the implementation/definition is not fully correct)
?: confusing or the idea is (somehow) good but not fully correct

Clarifications:
- Getting the length of an integer or extracting each digit is not a straightforward task. Therefore, not explaining (at least briefly) this/these process/es was penalized.
- Filtering if x is zero is not always necessary. If the way proposed to evaluate if each digit is not zero also (indirectly) checks for it, not having this initial check implicitly in the explanation is not a problem.
- If the module operation is mentioned but it is not specified which is the denominator of this operation, we penalise it very little (or even ignore it if the rest of the explanation is correct).
- Some of your suggested implementations do not scale properly when considering big values for x. For example, hardcoding if statements for x>10000, then x>1000, until x>0. We did not reduce much for this issue but consider that this is not a good practice (ideally, your implementation may work for any given value of x).

Note: Generally speaking, we were a bit stricter in the explanation of the cool numbers exercise as you have (almost) the whole code provided. For the rest, we tried to be more lenient.

Question 8 − Coding−Assignment − CODE − 149625.3.2

In this exercise, you must implement an algorithmic approach to what we call "cool" numbers. Given a (non-zero) positive integer `n`, a number `x` is considered "cool" if `n` is divisible by each of the number's digits (of `x`) without remainder.
Some examples follow:

- If n=16, then 2418 is a "cool" number because 16/2=8, 16/4=4, 16/1=16, 16/8=2 (none of these divisions leaves a remainder).
- If n=16, then 489 is not a "cool" number because 16/9=1 and leaves a remainder of 7. Note that the other digits are divisible with no remainder, however, recall that we want all of them to be divisible.
- If n=5, then 10000 is not a "cool" number since division by 0 is not possible -you need to take care of that in your program.

Complete the method below in such a way that given any positive integer (let that be `x`) and an integer `n` determines whether the number is "cool" or not (this means the method will return a boolean). This method returns true if the input a is "cool", and false otherwise.

```
public static boolean  isCool(int x, int n) {
    if (x==0)
        return false; ;
    int remaining =  x ;
    int nextDigit= remaining%10 ;           //Holds the next digit

    while (remaining !=0  ) {
        //once i find a zero or a non-divisible digit, i return false
        if ((nextDigit==0)  ||  (n%nextDigit!=0))
            return false ;

        remaining =  remaining/10 ;
        nextDigit =  remaining%10 ;
    }
return true ;
}
```

[Alphanumeric]	[Alphanumeric]	[Alphanumeric]	[Alphanumeric]	[Alphanumeric]
boolean	**return false;**	**x**	**remaining%10**	**!=0**
Boolean	return false		remaining%10;	!= 0
	return false;		remaining%10	!=0
	return false;		remaining%10	!= 0
			remaining%10;	!=0
			remaining%10;	
			x%10	
			x%10	
			x%10	
			x%10	

[Alphanumeric]	[Alphanumeric]	[Alphanumeric]	[Alphanumeric]	[Alphanumeric]
||	**return false**	**remaining/10**	**remaining%10**	**return true**
||	return false;	remaining/10;	remaining%10;	return true;
| |	return false;	remaining/10	remaining%10	return true
||	return false	remaining/10	remaining%10	return true
| |		remaining/10;	remaining%10;	return true;
| |		remaining/10;	remaining%10;	return true;

Question 9 − Coding−Consecutive − EXPLANATION − 149858.2.4

As owners of a data analysis company, we have received a task from one of our customers. This customer has a database that contains sequences of numbers that are not necessarily ordered consecutively. This is not a problem for our customer, but it makes an important task difficult, which is finding the longest consecutive sequence (e.g. 0, 1, 2, 3, 4, 5, etc.) in this database. The database is organized as a one-dimensional array. Our task is to implement a Java method that allows us to find the longest sequence of numbers in this database that starts with 0 and includes numbers in ascending order one by one.

Below, you can find some examples where all the sequences are indicated *in italics* and separated to improve visibility and the longest one is highlighted in **bold letters**.
Example 1:
Given: {*0, 1,* **0, 1, 2, 3, 4, 5**, 0, 5, *0, 1, 2, 3*}
The output is: 6

Example 2:
Given: {1, **0, 1, 2, 3**}
The output is: 4

Example 3:
Given: {1, 1, 1, 2, 3}
The output is: 0

In this case, even though there is a consecutive set of values (1, 2, 3), it does not start from 0. Additionally, as we do not have any 0, the longest one is 0.

Example 4
Given: {1, 1, 1, 2, 3, **0**}
The output is: 1
This is similar to the previous one but, in this case, as we have one element with the value 0, the longest one is 1.

Hint: It can be solved with recursion but you can also do it with loops.

Write IN YOUR OWN WORDS, how you would solve the problem. Do not use pseudocode but a high-level explanation of how you plan to solve the problem.
IMPORTANT: This part is only evaluated if you also provide a "good" attempt to implement it in the next question (coding part).

Grading instruction

Criterion 1 (Number of points: 10)

Grading criteria:
For full grade, we expect that the explanation includes:
- correct initialization of the counters
- loops for iterating over the main array
- considering 3 cases as shown in examples (no 0's in the sequence, only one 0, multiple 0's) in the explanation
- resetting counters
- condition on 0
- tracking the sequence starting at 0 (using a loop or counter variables)
- assigning max count
- no important syntax/conceptual errors found

Potential solution: see the attached Java files (one loop-based and one recursive solution)

Question 10 − Coding−Consecutive − CODE − 149702.2.3

As owners of a data analysis company, we have received a task from one of our customers. This customer has a database that contains sequences of numbers that are not necessarily ordered consecutively. This is not a problem for our customer, but it makes an important task difficult, which is finding the longest consecutive sequence (e.g. 0, 1, 2, 3, 4, 5, etc.) in this database. The database is organized as a one-dimensional array. Our task is to implement a Java method that allows us to find the longest sequence of numbers in this database that starts with 0 and includes numbers in ascending order one by one.

Below, you can find some examples where all the sequences are indicated *in italics* and separated to improve visibility and the longest one is highlighted in **bold letters**.
Example 1:
Given: {*0, 1,* **0, 1, 2, 3, 4, 5**, 0, 5, *0, 1, 2, 3*}
The output is: 6

Example 2:
Given: {1, **0, 1, 2, 3**}
The output is: 4

Example 3:
Given: {1, 1, 1, 2, 3}
The output is: 0
In this case, even though there is a consecutive set of values (1, 2, 3), it does not start from 0. Additionally, as we do not have any 0, the longest one is 0.

Example 4
Given: {1, 1, 1, 2, 3, **0**}
The output is: 1
This is similar to the previous one but, in this case, as we have one element with the value 0, the longest one is 1.

Hint: It can be solved with recursion but you can also do it with loops.

Use the following structure (method signature) for the method to be implemented:

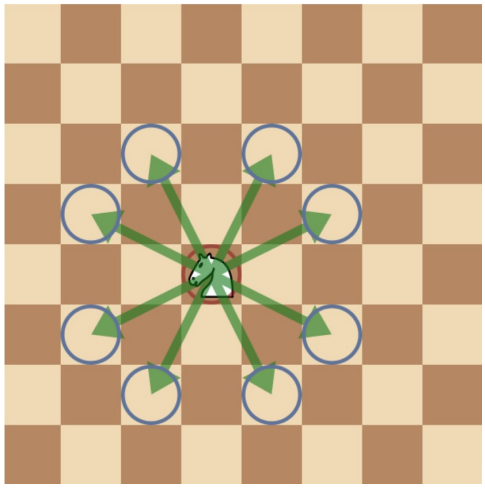
```
public static int consecutiveNumbers(int[] nums) { ... }
```

Grading instruction

Criterion 1 (Number of points: 10)

Question 11 − Coding−Chess − EXPLANATION − 149624.2.4

A chess knight can move in a shape that is typically described as an "L": two squares in one direction, and one square in a direction perpendicular to the previous one. The following figure shows where a knight can go in one move when it is in the middle of the board:

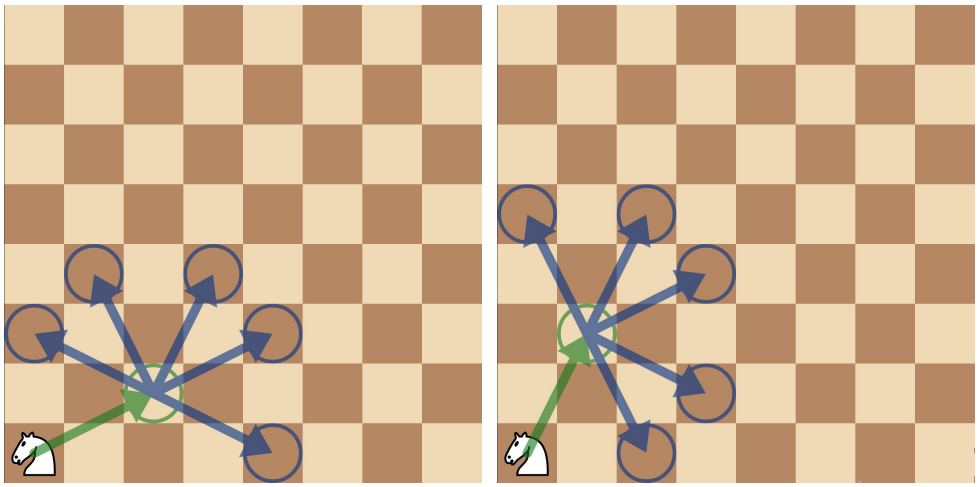


Naturally, a knight cannot move outside of the chess board. A knight that is closer to the edge of the board will have

restricted movement to the bounds of the board.

Considering an empty chess board (8x8), you are tasked with writing a method `isReachableInTwoMovements` that checks whether a knight, initially placed at a square given as two integers `startingRow` and `startingCol`, can reach the objective square defined also with two integers `objectiveRow` and `objectiveCol` **using at most two moves** (i.e. either one OR two movements). The method should return a *boolean* indicating whether the knight can reach the desired objective square in two moves or less.

Examples:



The white knight can reach all the marked squares in two movements or less.

Assuming the origin to be in the top left corner, the following statements:

- `isReachableInTwoMoves(7, 0, 7, 0)`

return TRUE (because the knight is on that square).

- `isReachableInTwoMoves(7, 0, 6, 2)`
- `isReachableInTwoMoves(7, 0, 5, 1)`

return TRUE (because the knight can move there in one move, marked in green in the images).

- `isReachableInTwoMoves(7, 0, 5, 0)`
- `isReachableInTwoMoves(7, 0, 4, 1)`
- `isReachableInTwoMoves(7, 0, 4, 3)`
- `isReachableInTwoMoves(7, 0, 5, 4)`
- `isReachableInTwoMoves(7, 0, 7, 4)`
- `isReachableInTwoMoves(7, 0, 3, 0)`
- `isReachableInTwoMoves(7, 0, 3, 2)`
- `isReachableInTwoMoves(7, 0, 6, 3)`
- `isReachableInTwoMoves(7, 0, 7, 2)`

return TRUE (because the knight can move there in two moves, marked in blue in the images).

Any other statements with origin startingRow=0, startingCol=0 would be FALSE.

Hint: It can be solved with recursion but you can also do it with loops.

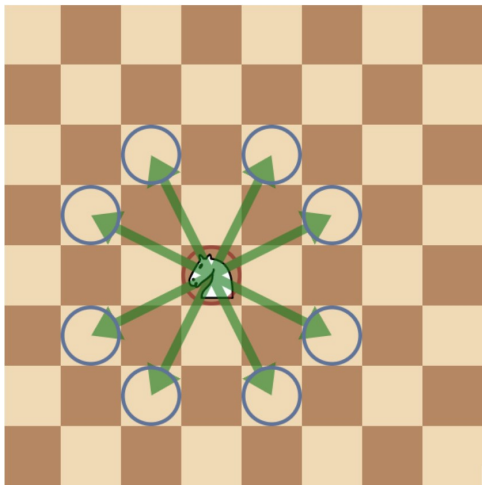
Write IN YOUR OWN WORDS, how you would solve the problem. Do not use pseudocode but a high-level explanation of how you plan to solve the problem.
IMPORTANT: This part is only evaluated if you also provide a "good" attempt to implement it in the next question (coding part).

Grading instruction

Criterion 1 (Number of points: 10)

Question 12 – Coding–Chess – CODE – 149922.2.6

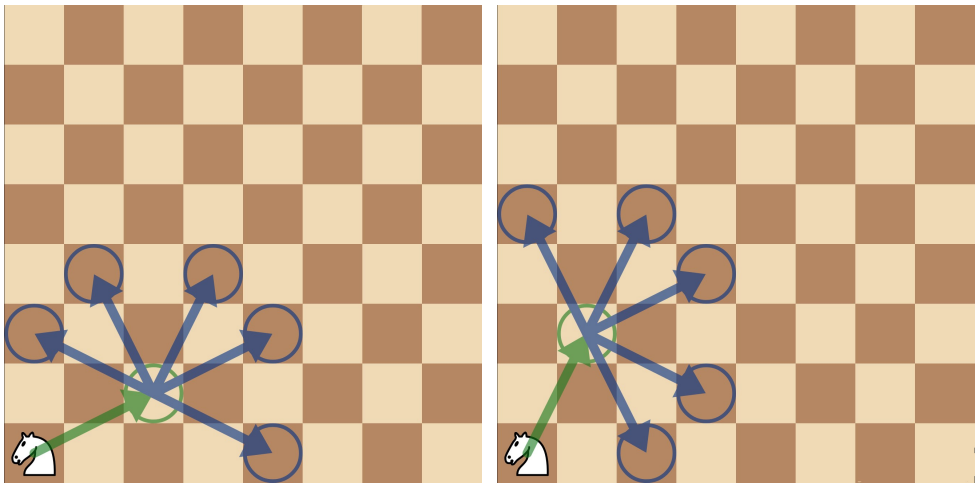
A chess knight can move in a shape that is typically described as an "L": two squares in one direction, and one square in a direction perpendicular to the previous one. The following figure shows where a knight can go in one move when it is in the middle of the board:



Naturally, a knight cannot move outside of the chess board. A knight that is closer to the edge of the board will have restricted movement to the bounds of the board.

Considering an empty chess board (8x8), you are tasked with writing a method `isReachableInTwoMovements` that checks whether a knight, initially placed at a square given as two integers `startingRow` and `startingCol`, can reach the objective square defined also with two integers `objectiveRow` and `objectiveCol` **using at most two moves** (i.e. either one OR two movements). The method should return a *boolean* indicating whether the knight can reach the desired objective square in two moves or less.

Examples:



The white knight can reach all the marked squares in two movements or less.

Assuming the origin to be in the top left corner, the following statements:

- `isReachableInTwoMoves(7, 0, 7, 0)`

return TRUE (because the knight is on that square).

- `isReachableInTwoMoves(7, 0, 6, 2)`
- `isReachableInTwoMoves(7, 0, 5, 1)`

return TRUE (because the knight can move there in one move, marked in green in the images).

- `isReachableInTwoMoves(7, 0, 5, 0)`
- `isReachableInTwoMoves(7, 0, 4, 1)`
- `isReachableInTwoMoves(7, 0, 4, 3)`
- `isReachableInTwoMoves(7, 0, 5, 4)`
- `isReachableInTwoMoves(7, 0, 7, 4)`
- `isReachableInTwoMoves(7, 0, 3, 0)`
- `isReachableInTwoMoves(7, 0, 3, 2)`
- `isReachableInTwoMoves(7, 0, 6, 3)`
- `isReachableInTwoMoves(7, 0, 7, 2)`

return TRUE (because the knight can move there in two moves, marked in blue in the images).

Any other statements with origin startingRow=0, startingCol=0 would be FALSE.

Hint: It can be solved with recursion but you can also do it with loops.

Use the following structure (method signature) for the method to be implemented:

```
public static boolean isReachableInTwoMoves(int startingRow, int startingCol, int objectiveRow, int objectiveCol)
```

Grading instruction

Criterion 1 (Number of points: 10)

Grading criteria:

- If only one movement is considered, the maximum grade is 3
- A good attempt (you are considering two movements but there are some small issues such as missing one input parameter in the recursive solution) would grade between 3 and 7.
- To obtain full grades, you need to consider the 2 possible movements with no, or very minor, mistakes
- Conceptual mistakes are penalized with1-2 points in most of the cases (unless it is an important one)

Potential solution: see the attached Java files (one implementation per each lecturer)