THE UNIVERSITY OF MANITOBA
COMP 1020: Introductory Computer Science 2
Term Test 2 (TT2)
A01 – Thursday, November 20, 2025

Student Information: Fill in one letter or number per box.

Academic Integrity Contract

I understand that cheating is a serious offence.

> *"As members of the University Community, Students have an obligation to act with academic integrity. Any Student who engages in Academic Misconduct in relation to a University Matter will be subject to discipline*." (2.4 - Student Academic Misconduct Procedure).

Student Signature:_____

Instructions – Please read carefully

1. Answer all questions on this paper ***in the spaces provided***. Answers must be on the same page as the question. Material on the cover page, a blank page, or on the wrong page, ***will not be marked***.
2. Answer questions using **Java programming language** and use **appropriate technical terms**.
3. Tests will be scanned for grading. **Write darkly**, and do not write very close to the edge of the paper, or in the **top** margin.
4. No aids ***of any kind*** (such as calculators, language translators, phones, smart watches, etc.) are permitted.
5. You have 60 minutes to complete the test.
6. There are 9 questions, and a total of 30 marks.
7. This test is worth 11% of your final grade in COMP 1020.
8. Fill in your identification information above. ***Do not*** write this information on any other page.

Part 1: Short Answer Questions (14 marks)

1. [3 pts total] For each of the following statements, indicate if the statement is True or False AND do the following.

- Correctly identify: **a True statement (1 mark), a False statement (0.5 marks).**
- **If the statement is False, indicate why (0.5 marks)**. You do not need to rewrite the entire statement; you can cross out words and replace them with the correct words or add words to the statement. You do not earn any marks for simply negating the statement (e.g., changing "This is a question." to "This is not a question.").

a) [1 pt] When locating a target in an unsorted list, binary search is more efficient than linear search.

False. Linear search is more efficient with unsorted data, as you would need to sort first and then do a binary search.
-0.25 if student only mentions you can't do binary search on unsorted data. It does not provide correct reasoning for question or make statement true. Should have indicating that sorting+binary search would be less efficient.

b) [1 pt] **`myArrayList.update(1,"words");`** updates the element at index 1 in an ArrayList.

False. The correct method would be myArrayList.set(1, "words");

c) [1 pt] Insertion sort continuously swaps elements into the correct location of an array to sort.

False. Selection sort continuously swaps elements; Insertion short 'shifts' elements until it finds the correct place to insert the value.

2. [1 pt] Other than the lack of comments, identify at least two programming standard violations in the code below.

```
public class Example {
    public static void Main(String[] args) {
    System.out.println( "Hello");
        if (true) {
            System.out.println("World"); }
    }
}
```

- Inconsistent brackets
- Inconsistent whitespace/spacing
- unnecessary if
- "true" and Strings could be considered literals/magic numbers

3. [2pt] What is the output of the code below?

```
public class Main {
    public static int process(int i, int j) {
        if (j <= 0) return 2;
            return i + process(i + 1, j - 1);
    }
    public static void main(String[] args) {
        System.out.println(process(5, 3));
    }
}
```

- 0.5 correct understanding of parameters on each call
- 0.5 correct understanding of how many times the function is called / hitting the base case
- 0.5 correct understanding of creating the result from the return variables
- 0.5 correct result

- minus 0.25 for 1 small error (like copied a wrong number and then used it for calculation)

Calls: process(5,3) → 5 + process(6, 2) → 6 + process(7, 1) → 7 + process(8,0) → return 2
Returns: 2 + 7 + 6 + 5 = 20
Output is **20**

4. [2 pts] Describe what the following code is doing and name the data structure it belongs to.

```
public void someMethod() {
    if (head != null && head.next != null) {
        head.next = head.next.next;
    }
}
```

```
Code is working with a LinkedList data structure
The method removes the second element of a linkedlist
- 1 correctly explains what it is doing...
- 1 recognize LinkedList and speaks about LL terms in the correct way (head, node, next, top, ll)
or:
- 0.5 for limited or literal explanation but does not identify that something is removed
- 0.5 for partially understanding it is LinkedList code (or not additionally explaining what it does)
```

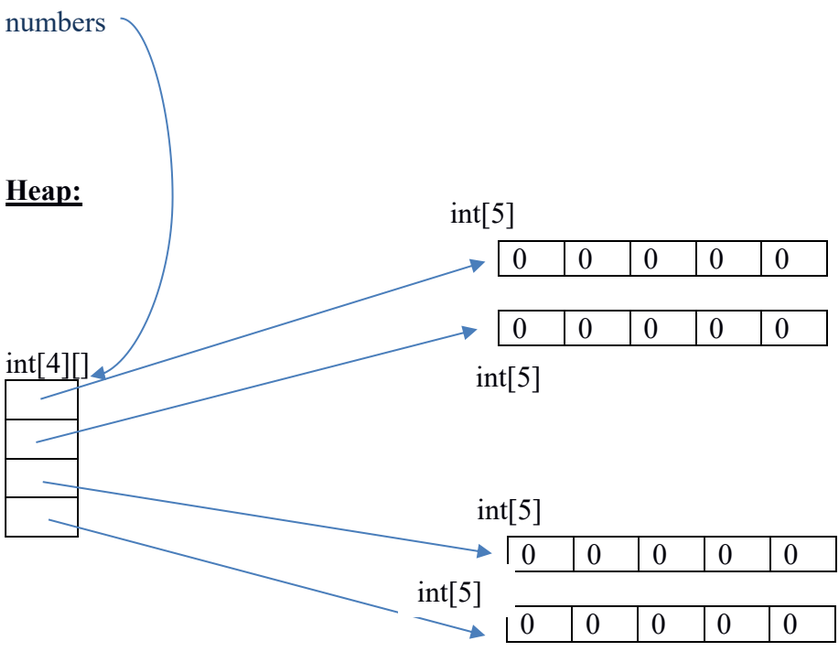
5. [2 pts total] Consider the following 2D integer array declaration:

```
int[][] numbers = new int[4][5];
```

a) [0.5 pts] How many total int values can be stored in this array?_ ***4 x 5 = 20 values (and indices)***_

b) [1.5 pts] Draw the memory diagram/representation of the multidimensional array. Include stack, heap, array(s) as objects, reference arrows, and any default values.

Stack:



0.5 correct representation of first array with pointers to other arrays
0.5 correct set of single arrays pointed to by first array
0.5 correct stack/heap components, other arrows, default values
-0.25 each small error
Common errors: the word "numbers" appearing in heap. Missing first array. Missing default values. Including too many labels for ***int[]*** *or* ***numbers[]*** *per index. Indices can exist on their own. Labeling a box of an array, but the box in the array should contain whatever value/reference pointer is stored there. Label the beginning of the array with type and size like "int[5]"*

6. [3 pts total] Answer the following two questions given about the code below:

```
public class Main {
    public static void sort(int[] arr) {
        for (int i = 1; i < arr.length; i++) {
            int j = i - 1;
            int target = arr[i];
            while (j >= 0 && arr[j] > target) {
                arr[j + 1] = arr[j];
                j--;
            }
            arr[j + 1] = target;
        }
    }
}
```

a) [1 pt] Name the sorting algorithm used in the code.

Insertion Sort *(see, no swapping code!)*

b) [2 pts] Demonstrate how the algorithm works by providing a step-by-step walkthrough of sorting the array: [5, 7, 3, 6, 1].

Ensure that your diagram captures the state of the array at the end of each iteration of the outer for loop (i.e., each 'pass' of the algorithm).

Start with: [5,7,3,6,1]

1st pass: [5,**7**,3,6,1] // 7 doesn't move
2nd pass: [**3**,5,7,6,1] // 3 moves to front

3rd pass: [3,5,**6**,7,1] // 6 is sorted

4 iteration: [**1**,3,5,6,7] // 1 moves to front and we are done

No 5th iteration.

-0.25 if missing last iteration, -0.25 if it is unclear what result is after each outer loop iteration,

-0.25 if correct iteration sort but does not match code

7. [1 pt] Which of the following is true about merge sort? Circle or indicate all that apply:

A. Two unsorted arrays are provided to the merge step. (needs sorted arrays for this)

B. Merge sort is more efficient than other sorting algorithms we have learned so far.

C. Merge sort can be implemented iteratively or recursively.

D. Merge sort is typically coded as an in-place algorithm. (no, we have to create more arrays)

0.5 for each right answer (B, C)

-0.25 for each wrong answer (A, D)

Part 2: Long Answer Questions (16 marks)

8. [8 pts total]

a) [5.5 pts] Write a recursive solution to determine if characters in a String are a palindrome – appear in the same order forwards and backwards. Your solution should ignore any whitespace and modify input to handle different letter cases (i.e., 'a' and 'A' should be considered as the same character). Follow best programming standards including method declaration, variable naming, and handling invalid input.

Example:

Input: "Hello" Result: false

Input: "RaceCar" Result: true

Input: "H a n na h " Result: true

```java
public static boolean isPalindrome(String word) {
    if(word == null) return false; // Guard clause, cannot proceed with null
    word = word.replaceAll("\\s+", "").toLowerCase();
    return isPalindromeHelper(word);
}
private static boolean isPalindromeHelper(String word) {
    // base case:
    // if we are looking at 1 letter or empty, we are done, return true
    if(word.length() <= 1) return true;

    // recursive case: check first and last characters of String for equality
    if(word.charAt(0) == word.charAt(word.length()-1)) {
        // these two match, continue inward
        return isPalindromeHelper(word.substring(1, word.length()-1));
    }
    return false; // not a palindrome, end processing
}
```

```
// note recursive question grading rubric is directly related to the question that was asked on each test and
varies slightly depending on which question you did.
// as mentioned; this question was marked 'stricter' than MD array question and error list in rubric to help
identify common mistakes that will lose points on tests / final exams

1.0 Method Signature:
public/private, static, boolean, clear name, String parameter
-0.25 if helper method has exact same signature(would not compile)
 or not private
-0.5 for incorrect or missing method line component
1.0 String processing:
correctly removes whitespace and ignores CASE
1.5 Base Case:
- returns false for null (0.5) // does not need to be in private helper
- returns true for String length() 0 (0.5) AND 1 (0.5)
-0.5 if returns something other than a boolean
-0.5 for bad base case check (will it actually reach?)
2.0 Recursive Case:
- checks if outer two chars are equal (0.5) correctly
- if true: returns recursive call (0.5) with smaller version of String (on each end) (0.5)
- if false: returns false (0.5)
-0.5 if results in infinite recursion (such as, did not update indices or make String smaller)

General:
    // Must be recursive
    // - NHT (no higher than) 2.0 if not recursive solution
    // -1.0 if wrote different recursive solution than asked / solved the problem w/o recursion
    // -0.25 for small syntax errors
    // -0.5 for errors
    // -0.5 for extraneous/unnecessary processing
    // -0.5 for repeating code
    // -0.5 other programming standard issues
```

b) [2.5 pts] Provide the input (parameters) to your method which will result in the method being called 3 times (total), including the first time with the original input (i.e., depth = 3). Show what the input and output is for each call. Call #3 should result in the base case being triggered.

Call #1 input: ____ "racecar"__(incorrect example)___

// in this code, the **private helper** method is the recursive method

Call #2 input: ____ "aceca"_____

Call #3 input: ____ "cec" ___// whoops this is incorrect! Should have started with input "aceca" as #1 input to be able to reach the input that will trigger the base case "e"

Base Case Reached!

// pretend we started correctly and output 3 is coming out of the base case

Output #3: ___true // returned from base case from "e" input

Output #2: ____ true // returned from call with "cec"

Output #1: _____ true _// returned from call with "aceca" input ___ **(final result)**

9. [8 pts] Write a method that will be passed in a 2D array of Seat objects. Each row represents a row on an airplane. You may assume:

   • The 2D array has been filled elsewhere (there are no null values)

   • Each row has an even number of seats, though rows may vary in size.

   • Each row has at least two seats

   • Seat has the method: boolean isOccupied()

Your method should output the total number of occupied seats as a percentage, as well as the number of aisle/window seats that are occupied. Window seats are the first and last seats of a row, and aisle seats are the two middle seats of a row.

The method should print the following:

Percentage occupied: 90%
Aisle seats occupied: 9
Window seats occupied: 12

Rubric: 8 points

1.0 – // Method setup:

- Correct access modifier, static/non, return type, parameters (correct 2D array)

4.0 // MD array

- Correct access of array and indices
- Correct nested loops and iterating through 2D array
- Correct lengths for inner loop for ragged arrays
- Correct understanding of rows/cols with aisle on end of row, seats in the middle of row

2.0 // Building result

- Correctly identifying and saving total number of seats
- Correctly identifying and tracking isOccupied
- Correctly counting windows / Correctly counting aisles
- Correct calculation of result and creating output String

1.0 // Correct Result returned

Blank Page for continued work or scrap notes.

If you are continuing work, clearly indicate which question is continued on this page and where.

Use this page for rough work only. It will NOT be graded.