Ghulam Ishaq Khan Institute of Engineering Sciences and Technology
Faculty of Computer Science and Engineering

CS 112 Section H

Vetted by Dr. Hanif [illegible]

Spring 2024 Semester

Final Examination

Monday 20-05-2024

Total Time: 150 minutes

Total Marks: 75

Course Instructor:

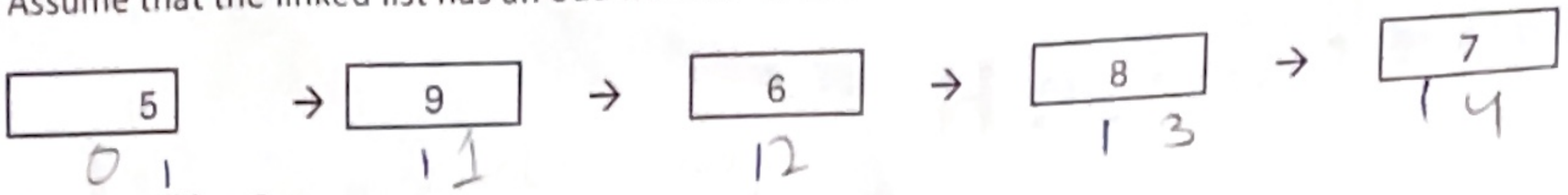
Prof. Dr. Zahid Halim

You are advised to READ these notes:

1. Attempt all the questions on the provided answer sheet only.
2. There are a total of eight (8) questions including one bonus.
3. Understanding the question in part of the examination.
4. **The invigilator present is not supposed to answer any questions. No one may come to you for corrections, and you are not supposed to request to call anyone. Make assumptions wherever required and clearly mark them.**

Best of luck!

Q1: [10 points, difficulty level=Medium, CLO-2] Write a function that receives the head pointer of the linked list as a parameter and displays the value stored in the middle node of the linked list. The **restriction** is that you have to use only one loop in your code, and the loop can be iterated only once. Assume that the linked list has an odd number of nodes. For example, for the following linked list:



Output will be: 6

Q2: [10 points, difficulty level=low, CLO-1] Package-delivery services, such as FedEx®, DHL® and UPS®, offer a number of different shipping options, each with specific costs associated. Create an inheritance hierarchy to represent various types of packages. Use Package as the base class of the hierarchy, then
base
include classes TwoDayPackage and OvernightPackage that derive from Package. Base class Package
derive
should include data members representing the name, address, city, state and ZIP code for both the sender and the recipient of the package, in addition to data members that store the weight (in ounces) and cost per ounce to ship the package. Package's constructor should initialize these data members. Ensure that the weight and cost per ounce contain positive values. Package should provide a public member function calculateCost that returns a double indicating the cost associated with shipping the package. Package's calculateCost function should determine the cost by multiplying the weight by the cost per ounce. Derived class TwoDayPackage should inherit the functionality of base class Package, but also include a data member that represents a flat fee that the shipping company charges for two-day-delivery service. TwoDayPackage's constructor should receive a value to initialize this data member. TwoDayPackage should redefine member function calculateCost so that it computes the shipping cost by adding the flat fee to the weight-based cost calculated by base class Package's calculateCost function. Class OvernightPackage should inherit directly from class Package and contain an additional data member representing an additional fee per ounce charged for overnight-delivery service. OvernightPackage should redefine member function calculateCost so that it adds the additional fee per ounce to the standard cost per ounce before calculating the shipping cost. Write a test program that creates objects of each type of Package and tests member function calculateCost.

Q3: [10 points, difficulty level=Low, CLO-2] Design a class ***StringTransformer*** that provides various functionalities to manipulate and analyze strings. The class should have the following methods:

- ***init()***: Initializes the object with the string s.
- ***is_palindrome()***: Checks if the string s is a palindrome.
- ***longest_repeating_substring()***: Finds and returns the longest substring that appears at least twice in the string s.

Constraints:

- The length of the input string will be at most 1000.

Requirements:

- The class should be well-encapsulated, with proper use of private and public methods.
- Use appropriate class attributes to store the string and any other necessary data.

Ensure that the methods are efficient and follow OOP principles.

Q6: [10 points, difficulty level=low, CLO-1] Create a class called Complex for performing arithmetic with complex numbers. Write a program to test your class.

Complex numbers have the following form:
realPart + imaginaryPart * i

where i is $\sqrt{-1}$

Use double variables to represent the private data of the class. Provide a constructor that enables an object of this class to be initialized when it is declared. The constructor should contain default values in case no initializers are provided. Provide public member functions that perform the following tasks:

a. Adding two Complex numbers: The real parts are added together and the imaginary parts are added together.
b. Subtracting two Complex numbers: The real part of the right operand is subtracted from the real part of the left operand, and the imaginary part of the right operand is subtracted from the imaginary part of the left operand.
c. Printing Complex numbers in the form (a, b), where a is the real part and b is the imaginary part.

Q7: [10 points, difficulty level=low, CLO-1] You are given a string containing only the characters '(', ')', '{', '}', '[', and ']'. Write a C++ function to determine if the input string is valid according to the following rules:

- Open brackets must be closed by the same type of brackets.
- Open brackets must be closed in the correct order.

You must implement the following function:
bool isValid(string s);

Input:
A string s containing only the characters '(', ')', '{', '}', '[', and ']', where 1 <= s.length <= 10^4.

Output:
Return true if s is valid, and false otherwise.

Examples:
```
isValid("()") // true
isValid("()[]{}") // true
isValid("(]") // false
isValid("([)]") // false
isValid("{[]}") // true
```

Note: An empty string is also considered valid.

Hints: Consider using a stack to keep track of the opening brackets. Assume the class Stack is already available, and you can use its push and pop functions by creating an instance of it.

Q8: Bonus IQ question [3] A farmer has 17 sheep, and all but nine die. How many sheep are left?

Q4: [12 points, difficulty level=Medium, CLO-2]

Consider the following sample data:

```
150
4

5.1   3.5   1.4   0.2
4.9   3     1.4   0.2
4.7   3.2   1.3   0.2
```

$$\frac{\Sigma(X_i-\bar{X})(Y_i-\bar{Y})}{\Sigma(X-Y)}$$

- ➢ The digit 150 in first row is the number of rows
- ➢ Digit 4 in second row is the number of columns
- ➢ Third row is an empty one
- ➢ Rest is a grid of data.

Write the following different functions for the above dataset:

- Write a function that reads the data into a 2-D array named **Data[Rows][Columns]**.
- Write a function that finds the correlation of each row with each of the other rows. The result is stored in a new 2-D array named **CorrelationMatrix[Rows][Rows]**. For computing the correlation, assume there is a built-in function float **correlation(float* data, int size)** which you simply need to call.
- Write a function that discretizes the **CorrelationMatrix** by computing the mean (i.e., average) of each column.

Q5: [10 points, difficulty level=medium, CLO-3]

You are required to implement a simplified version of the popular game "Battleship" using C++. The game should have the following features:

- A Board class representing the game board, which is a 10x10 grid. Each cell in the grid can either be empty or contain a part of a ship.
- A Ship class representing a ship, with attributes for size, position (row and column), and orientation (horizontal or vertical).
- A Player class representing a player, with a name and a board to place their ships on.
- A Game class representing the game itself, which contains two players and manages the game flow.

The game should proceed as follows:

- Players take turns to place their ships on their board. Ships cannot overlap or extend beyond the board boundaries.
- After all ships are placed, players take turns to guess the positions of their opponent's ships. The opponent responds with "Hit" if the guess hits a ship part, or "Miss" otherwise.
- The game continues until one player has all their ships sunk, at which point the other player wins.

Implement the necessary classes and functions to simulate the Battleship game. Use appropriate OOP concepts such as classes, objects, inheritance, and encapsulation.

Do NOT Write the main() function.