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

CS 112

Vetted by _____

Spring 2022 Semester

Approved by: _____

Final Examination

Past paper for reference only

Thursday 09-06-2022

Total Time: 10800 seconds

Total Marks: 80

Course Instructor:

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 seven (7) questions including one bonus.
3. **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.**

Enjoy!

Q1: [10] In computer science, a stack is an abstract data type that serves as a collection of elements, with two principal operations:

push, which adds an element to the collection, and
pop, which removes the most recently added element that was not yet removed.

The order in which elements come off a stack gives rise to its alternative name, LIFO (last in, first out). Additionally, a peek operation may give access to the top without modifying the stack. The name "stack" for this type of structure comes from the analogy to a set of physical items stacked on top of each other, which makes it easy to take an item off the top of the stack, while getting to an item deeper in the stack may require taking off multiple other items first.

There are two implementations of stacks, static stack and dynamic stack in context of memory management. Size of static storage area is constant throughout execution but dynamic stack grows and shrinks as per push and pop of activation record.

Write C++ code to make a class that represents a dynamic stack. The class should have functionality for pop, push, and peak, in addition to proper having a constructor and destructors.

Q2: [10] Given classes Circle, Square, and Triangle derived from a class Shape, define a function intersect() that takes two Shape* arguments and calls suitable functions to determine if the two shapes overlap. It will be necessary to add suitable (virtual) functions to the classes to achieve this. Don't bother to write the code that checks for overlap; just make sure the right functions are called. This is commonly referred to as double dispatch or a multi-method.

Q3: [15] In this question you will use programming to simulate bunnies. In our simulation, each bunny is on a grid at some location defined by an **X-coordinate** and a **Y-coordinate**. Also, each bunny has some number of energy units measured in carrot stick sticks. (X-coordinates, Y-coordinates, and the number of carrot sticks are integer values.)

- Bunnies can hop north, south, east, or west.
- When a bunny hops to the north, the bunny's Y-coordinate is increased by 1 and the X-coordinate remains unchanged.
- When a bunny hops to the west, the bunny bunny's X-coordinate is decreased by 1, and the Y-coordinate remains unchanged.
- When a bunny hops to the east, the bunny's X-coordinate is increased by 1 and Y-coordinate remain unchanged
- When a bunny hops to the south Y coordinate is decreased by 1 and X-coordinate remains unchanged.

Note that making one hop requires a bunny to eat one carrot stick, and when a bunny has eaten all of his or her carrot sticks, that bunny can 't hop.

Create a Bunny class which can be used to generate Bunny objects that behave as described above.

When a new Bunny object is created, the Bunny always starts at coordinates X = 10, Y = 10, and the Bunny has 5 carrot sticks.

Your **Bunny** class definition must include a **hop(int direction)** method. The direction parameter is **12 for north**, () **3 for east**, **6 for south**, and **9 for west** (like a clock face). The hop **hop()** method should test to make sure () that the Bunny has not eaten all the carrot sticks. If the Bunny still has carrot sticks, the **hop()** method should update coordinates as explained above and print the message **"hop"**. If no carrot sticks remain, it should just print the message **"This bunny can't hop"**.

Q4: [10] You are the owner of a hardware store and need to keep an inventory that can tell you what different tools you have, how many of each you have on hand and the cost of each one. Write a program that initializes the random-access file hardware.dat to 100 empty records, lets you input the data concerning each tool, enables you to list all your tools, lets you delete a record for a tool that you no longer have and lets you update any information in the file. The tool identification number should be the record number. Use the following information to start your file:

Record#	Tool name	Quantity	Cost
3	Electric sander	7	57.98
17	Hammer	76	11.99
24	Jig saw	21	11.00
39	Lawn mower	3	79.50
56	Power saw	18	99.99
68	Screwdriver	106	6.99
77	Sledge hammer	11	21.50
83	Wrench	34	7.50

Q5: [15] Write a class (main () NOT required) to simulate the game "Game of Guess". The concept of the game is, computer will randomly think of any country, it will display on screen underscores which will be equivalent to the length of the country name the computer has thought of.

For example in case computer thinks Japan then it will display _ _ _ _ _

The user will enter a single character at a time which s/he founds appropriate if the chosen character is not in the name of country user looses 1 chance out of 5 possible chances of mistake. Otherwise the chosen character is replaced in the underscore.

For example in case user enters **N** your program should display _ _ _ _ N

Constraints:

- Once a character is chosen it cannot be chosen again.
- Maximum number of mistakes is 5. But you have to provide a function in class to change it.
- Display the number of chances left for the user after each input character
- Provide functionality to reset the game i.e. chosen country, chances and display
- Provide a function that at the end of game shows the result i.e win/loose with the correct answer that will be the name of the country.
- Provide a function which can allow a user of class to put names of countries of his/her choice in the array

Q6: [20] TopiPhoneCard Inc. sells phone cards for making cheap long distance calls around the world. In this question, you will write a C++ program using classes to manage their business.

TopiPhoneCard Inc. sells 3 types of phone cards: SuperNA10 cards, which cost Rs.10 and are good only for calls in Pakistan and Afghanistan, Global10 cards, which cost Rs.10 and are good for overseas calls, and Global25 cards, which cost Rs. 25 and are also good for overseas calls. The per minute rates for each type of card and call zone are as follows:

	SuperNA10	Global10	Global25
Pakistan	Rs.0.05	Rs.0.07	Rs.0.05
Afghanistan	Rs.0.10	Rs.0.15	Rs.0.10
Europe	XXXXX	Rs.0.30	Rs.0.20
Asia	XXXXX	Rs.0.60	Rs.0.40
Australia & NZ	XXXXX	Rs.0.45	Rs.0.30
Latin America	XXXXX	Rs.0.45	Rs.0.30
Africa	XXXXX	Rs.0.60	Rs.0.40

The initial balance on the cards and the weekly maintenance fee are indicated below:

	SuperNA10	Global10	Global25
initial balance	Rs.10.00	Rs.10.00	Rs.25.00
weekly fee	Rs.0.50	Rs.1.00	Rs.1.00

Your main job in this question is to implement a hierarchy of classes to represent the different types of cards. At the top of the hierarchy, there is an abstract class PhoneCard with the following features:

- public PhoneCard(long no, int passwd, double bal): a constructor
- public long getNumber(): returning the card number;
- public int getPassword(): returning the card password;
- public double getBalance(): returning the card balance;
- public void setBalance(double bal): to set the card balance;
- public boolean allowed(CallZone zone): returns true if a call to the argument zone is allowed for the card;
- public double costPerMin(CallZone zone): returns the cost per minute of a call to the argument zone;
- public int getLimit(CallZone zone): returns the maximum number of minutes that can be charged for a call to the argument zone given the card's balance
- public boolean charge(int minutes, CallZone zone): tries to charge a call to the given zone with the given number of minutes to the card; if the balance is sufficient to cover it, it returns true and if the balance is not sufficient, it leaves it unchanged and returns false
- public void deductWeeklyFee(): deducts the appropriate weekly fees from the card's balance, leaving it non-negative;
- public String toString(): returns the string "card no *no* has a balance of *X.XX*".

There are also 3 classes, SuperNA10Card, Global10Card, and Global25Card that inherit from PhoneCard, and model the properties of the associated type of card as specified above, by defining the PhoneCard class's abstract methods.

Q7: Bonus question [5] Before Mount Everest was discovered, what was the highest mountain on earth?