CS224 Object Oriented Programming and Design Methodologies Lab Manual

Lab 0x - Practice Lab

Department of Computer Science

Dhanani School of Science and Engineering

Habib University

Fall 2025

Contents

1 Introduction

1.1 Guidelines

1. Use of AI is strictly prohibited. This is not limited to the use of AI tools for code generation, debugging, or any form of assistance. If detected, it will result in immediate failure of the lab, student will be awarded 0 marks, reported to the academic integrity board and appropriate disciplinary action will be taken.
2. Absence in lab regardless of the submission status will result in 0 marks.
3. All assignments and lab work must be submitted by the specified deadline on Canvas. Late submissions will not be accepted.

1.2 Objectives

Following are the lab objectives of this lab:

- To revise basic C++ concepts like loops and conditional statements.
- To get familiar with arrays in C++.
- To know how to apply operations on arrays in C++.

1.3 Directory Structure

Labs will have following directory structure:

```
lab
├── manual
│   └── lab.pdf
├── src
│   ├── *.hpp
│   └── *.cpp
├── tests (optional)
│   └── *.cpp
└── makefile (optional)
```

`manual` will contain the lab manual pdf. `src` will contain the source code files, and `tests` (if present) will contain the test files. `makefile` (if present) will contain the makefile for testing and running.

2 Exercises

2.1 Happy and Sad Numbers

Any positive number is called a **Happy Number** when we calculate the sum of squares of its digits on a repeated basis until the number becomes a single digit and that digit is 1, if that final digit is other than 1 then it is called unhappy or **Sad Numbers**.

Example 01 : Let's check 23
Step1: $23 = 2^2 + 3^2 = 4 + 9 = 13$ (sum of the squares of its digits)
Step2: $13 = 1^2 + 3^2 = 1 + 9 = 10$ (sum of the squares of the digits)
Step3: $10 = 1^2 + 0^2 = 1$ (sum of the squares of the digits)
It ends on 1, hence it is a happy number

Example 02: Let's check 36
Step 1: $36 = 3^2 + 6^2 = 9 + 36 = 45$
Step 2: $45 = 4^2 + 5^2 = 16 + 25 = 41$
Step 3: $41 = 4^2 + 1^2 = 16 + 1 = 17$
Step 4: $17 = 1^2 + 7^2 = 1 + 49 = 50$
Step 5: $50 = 5^2 + 0^2 = 25 + 0 = 25$
Step 6: $25 = 2^2 + 5^2 = 4 + 25 = 29$
Step 7: $29 = 2^2 + 9^2 = 4 + 81 = 85$
Step 8: $85 = 8^2 + 5^2 = 64 + 25 = 89$
Step 9: $89 = 8^2 + 9^2 = 64 + 81 = 145$
Step 10: $145 = 1^2 + 4^2 + 5^2 = 1 + 16 + 25 = 42$
Step 11: $42 = 4^2 + 2^2 = 16 + 4 = 20$
Step 12: $20 = 2^2 + 0^2 = 4 + 0 = 4$
The final digit is not 1 so this number is a Sad Number.

Task: Write the following program:
Define a function named `FindHappiness`, which takes an integer as a parameter and finds the happiness of the number. If the number is happy just print "Happy Number!" and if it is a sad number then print "Sad Number!".

Write the main function which takes an integer n as user input, if $n < 1$ then print an error message : "Invalid input! Try again!" then take the input again until a valid value is entered and finally passes that integer n to the function `FindHappiness()`.

Input Format For Custom Testing:
The first line will contain the integer that you need to check for - whether it is a happy or sad number,^).

Sample Case 0

```
>>> 23
Happy Number!
```

Sample Case 1

```
>>> 36
Sad Number!
```

2.2 Numpy Arrays in C++

In this question you will be implementing a very simple Numpy array as a class. It can store exactly 10 values in an integer array (declared as an attribute in the class).

As given in the main function comments, the Numpy objects are able to be taken input from cin, can be printed using cout, and can be added with other Numpy objects and an integer.

Input Format For Custom Testing:
The first line contains an integer, n, denoting the number of elements in `ARRAY_NAME`. Each line i of the n subsequent lines (where $0 \leq i < n$) contains $a(n)$ `DATA_TYPE` describing $\texttt{ARRAY\_NAME}_i$ The very last line will contain an integer that will be used for addition with the numpy array.

Sample Case 0

```
>>> 10
>>> 52 90 26 0 22 51 19 56 6 13
>>> 55 2 73 68 94 -5 10 56 12 6
>>> 50
n1 + n2 = [107, 92, 99, 68, 116, 46, 29, 112, 18, 19]
n1 + 50 = [102, 140, 76, 50, 72, 101, 69, 106, 56, 63]
```

Explanation: Element by element addition is done when n1 + n2. 50 is added with all the elements of n1.

2.3 Bank Account

Create a class BankAccount which has:

- accountTitle (type character string)
- balance, limit (type int)
- `deposit(int amount)`: this function deposits the given amount into the account balance.
- `withdraw(int amount)`: this function withdraws the given amount from the account. The balance and limit of the account should be more than the amount to be withdrawn.
- `BankAccount()`: A constructor that initializes the bank account with an empty character string for accountTitle and zero values for balance and limit.
- `BankAccount(string title, int bal)`: This constructor initializes the title and balance of account. limit is set to zero.
- `BankAccount(string title, int bal, int lim)`: This constructor initializes all the attributes as provided in the argument.

Main Function: You have to write the main function for this program as well. You'll be creating only one object here with three argument constructor, however, in the class you must define the other two constructors.

Input Format For Custom Testing:
The first line of the input gives the account title, initial balance, and daily limit values. Hence, you create an object of BankAccount with these three given values.

The proceeding lines define the transactions (a withdrawl, or a deposit), so you apply the transactions accordingly to the account object.

The last line contains the character q, where you stop reading the input.

Sample Case 0

```
>>> Usman 40000 10000
>>> w 15000
>>> d 30000
>>> q
Withdraw Failed: daily limit is 10000
Successfully Deposited 30000
Closing Status:
Title: Usman, Current Balance: 70000, Daily Limit: 10000
```

Explanation: Withdraw failed because the daily limit is 10000. Rs. 30000 are added to account, so now the balance is 70000.

Sample Case 1

```
>>> Farhan 40000 60000
>>> w 25000
>>> w 20000
>>> d 30000
>>> d 70000
>>> q
Withdrew 25000 successfully
Withdraw failed, insufficient funds
Successfully Deposited 30000
Successfully Deposited 70000
Closing Status:
Title: Farhan, Current Balance: 115000, Daily Limit: 60000
```

Explanation: Successfully withdrew 25000. Withdraw of 20000 failed because of insufficient funds. Successfully deposited 30000 and 70000, so now the balance is 115000.