

# Programming Assignment – 3

CS200 - Introduction to Programming

Dr. Mian Muhammad Awais

**Due Date: 25th November 2020, 11:55 PM**

**Total Marks: 150**

## **INSTRUCTIONS (Important):**

- Start the assignment early so that you can finish within the deadline. We have the following late submission policy: you have a pool of 5 grace days that you can utilize for all assignments with a 0% reduction in your marks. **Remember that there will be no extensions after you have utilized all the grace days**, so use them wisely across all assignments. Any submission past the deadline will count as having utilized a grace day, so do not wait for the last minute to submit to avoid late submission due to connectivity issues or any other issues.
- If you have used all your grace days, no late submission will be accepted
- No email submissions will be accepted. You must submit your solution on LMS
- You have to submit a .zip file named 23100XXX\_PA3.zip containing 2 .cpp files named: 23100XXX \_questionX\_PA3.cpp
- Do not submit ANY other files except your .cpp files. E.g.exe files, .cbp files, other project files, or any other files should not be present in your submission.
- We expect your programs to work on all valid inputs. If it fails to work on any valid input, you will be awarded half marks for that part. The examples given under questions are not exhaustive, be sure to check all corner cases
- The marks for every part of all questions have been written in square brackets.
- In case you have any queries, please feel free to post them on Piazza and ask the TAs during office hours.
- **Do not copy code from the internet. Do not share or look at any other student's code. If in doubt, talk to Sir or any TA. Strict action will be taken against plagiarism**

**Question 1 [50]:**

Board-Game scenario:

Consider a board game (Board: 2D Dynamic Array/ 2D Vector ( $m \times n$ )). The player starts the game at the top left corner of the board. While he ends the game at the bottom right corner. The board elements are either marked with 0 or 1. During the game, if any cell of the board is marked with zero, the player can't go to that cell. The player can only move to the board cell if the cell is marked with 1. Moreover, the player can only move to the consecutive right or down cell. You need to find if there is any possibility for the player to reach the bottom right corner of the board. Note: Top left cell always contains 1.

The required functions for this question are:

1. You need to implement the function **inputBoard** which inputs the board from the user in the 2D Dynamic Array / 2D Vector. The number of rows does not necessarily have to be equal to the number of columns. **[5]**
2. You need to implement another function **findPossibility**. This takes the board as input and states if there is any possibility for the player to reach the end of the board. **[45]**

**Works for no input, max marks possible = 10, based on code**

**If only works for some input, max marks possible = 30**

**Works for all inputs correctly = 45**

The prototype of function for 2D dynamic array should look something like this:

**bool findPossibility(bool\*\* board, int row, int col);**

Sample Input and Output:

Input Example:

```
arr[][] =  
{  
    {1, 1, 1, 0, 1},  
    {0, 1, 1, 0, 0},  
    {1, 1, 1, 0, 1},  
    {0, 1, 1, 1, 1},  
    {1, 1, 0, 1, 1}  
}
```

Output: True

Input Example: arr[][] =

```
{  
    {1, 1, 1, 0, 1},  
    {0, 1, 1, 0, 0},  
    {1, 1, 1, 0, 1},  
    {0, 1, 0, 1, 1},  
    {1, 1, 0, 1, 1},  
    {0, 0, 0, 0, 1}  
}
```

Output: False

## **Question 2 [100 marks]:**

In this next task, you will need to implement a program using class inheritance:

There will be one class named '**Clock**' with attributes: **[5]**

1. Minutes
2. Hour
3. Meridiem (will store 'AM' or 'PM')

And methods:

1. `display_time()` **[5]**

You are also required to make the default and parameterized constructors along with getters and setters.

Another class, '**Clock\_angle**' will inherit Clock publicly and contains the following variables and member functions: **[5]**

Attributes:

1. Angle

Methods:

1. `calculate_clock_angle()`
2. `display_time()` **[20]**

Make the default and parameterized constructors and setter and getters

**Note:** Both `display_time` functions will have the same prototype, but their functionality will be different. The `display_time()` function of class 'Clock' will display the time as 'hh: mm' while in `clock_angle` it will display the time in 24hr format 'XXXX hrs'

### **Calculate\_clock\_angle(): [35]**

This function will determine the size of the angle between the hour and minute hands of the clock object.

It will also take an argument: 'minutes\_elapsed' and calculate the angle between the clock hour hand and the new minutes. New minutes = clock minutes + minutes\_elapsed

(You may assume for simplicity that minutes\_elapsed will range from 0 to 60. Make sure to deal with the corner cases accordingly then)

The function should **print both the above-mentioned angles to the console and return the difference of clock-hour-clock-minutes angle subtracted from new\_minutes-clock-hour angle.**

#### **Note:**

Assume all angles are positive and consider the smaller of the two possible angles between the hands. For E.g., if the hour = 9 and minutes = 0, the angles between the two hands are 90 degrees and 270 degrees, but you will only consider the 90-degree angle.

In the main function

Make a pointer to the clock class and an object of clock\_angle and make the pointer of the base clock class point to the clock\_angle object and use the pointer to call the necessary functions where possible. [10]

You are required to prompt the user for the time, which will be entered as "hhmm" (i.e., 12-hour format). You must do all error handling to ensure that a valid time is entered. [10]

E.g

1234 is valid

2314 is not valid

959 is valid

1060 is valid, but you need to interpret it as 1100

Keep prompting the user for a valid input if an invalid time is entered

And then prompt for 'AM or PM'

After entering the time, display the time in 12-hour format then in 24-hour format and then call calculate\_clock\_angle() which would print the 2 angles (make sure to include prompts to specify which angle between which hands is being printed) and then print the returned angle difference

### **Correct output [10]**

Sample output:

If input = 900, AM and angle parameter in calculate\_clock\_angle() = 15

Time in 12-hour format: 9:00 AM

Time in 24-hour format: 0900 hrs

The angle between clock hour and clock minute: 90 degrees

The angle between clock hour and minutes elapsed: 180 degrees

Angle difference: 90 degrees

### **Marks Distribution details:**

1. If any constructor or setter or getter missing, award 0/5 marks for either class structure
2. If 24-hour display\_time() works for values and not for other, award 10/20 marks
3. Calculate\_Clock\_angle()
  - a. If larger angles considered instead of smaller ones, max marks = 20 (with everything else correct)
  - b. If some angles correct and some wrong, award 20/35
  - c. If no angles correct, max marks = 10/35
  - d. If all angles correct, award 35/35
4. If pointer for base class not used with inherited object, 0/10
5. Validation of inputs:
  - a. If all inputs correctly validated, 10/10
  - b. If 60 minutes not handled, 5/10
  - c. If some invalid input regarded valid or vice versa, 5/10

6. Output format:

- a. If description for values missing, 5/10
- b. One item missing, 5/10
- c. Two or more items missing from output, 0/10