

Project - 1 MA517M-Basic Programming Laboratory Last Date: 14 November 2026
Name Roll No.: MA26M012

Integer Partitions using C++ Classes and Operator Overloading

Objective: To design a C++ program that implements integer partitions using classes, generates all partitions, counts them, and supports comparison using operator overloading.

A partition of a positive integer n is a way of writing n as a sum of positive integers, without considering the order of the summands. For example, the partitions of 4 are:

$$4, 3 + 1, 2 + 2, 2 + 1 + 1, 1 + 1 + 1 + 1$$

Problem Description

Design a class `IntegerPartition` to represent partitions of a positive integer. The class should generate all partitions, count the total number of partitions, access individual partitions, and support comparison between partitions using operator overloading.

Class Specification

- **Class Name:** `IntegerPartition`
- **Private Data Members:**
 - `int n;`
 - `vector<vector<int>> partitions;`
- **Public Member Functions:**
 - `IntegerPartition(int x);` Constructor initializes partitions of x
 - `void generate();` Generates all partitions of n
 - `int count() const;` Returns the total number of partitions
 - `vector<int> getPartition(int k) const;` Returns the k -th partition
 - `void display() const;` Displays all partitions

Operator Overloading

- `operator ==( ), !=( )` Compares two partitions (exact equality of all summands)
- `operator []` Accesses the k -th partition
- `operator <<( )` Displays all partitions in a readable form

Tasks

1. Create an `IntegerPartition` object for $n = 5$
2. Generate all partitions and display them
3. Count the total number of partitions
4. Access and display the 3rd partition using overloaded operator `[]`
5. Compare two partition objects for equality

Expected Output Example

Partitions of 5:

```
5
4 + 1
3 + 2
3 + 1 + 1
2 + 2 + 1
2 + 1 + 1 + 1
1 + 1 + 1 + 1 + 1
```

Total number of partitions = 7

3rd partition = 3 + 2

Comparison of Partition1 and Partition2: False

Project - 2: Match-3 Game (Candy Crush Variant) Using C++ Classes

Problem Statement

Design and implement a **Match-3 Game** using **C++ classes**. The game consists of a grid of colored tiles (or symbols), where the player swaps adjacent tiles to form a sequence of three or more identical tiles in a row or column. When a match is formed, the tiles disappear, points are awarded, and new tiles fall from the top to fill empty spaces. The project should utilize object-oriented programming concepts such as classes, objects, encapsulation, and methods to handle grid updates, scoring, and user interaction.

Project Requirements

1. Create a `Tile` class to represent an individual tile in the grid.
 - (a) Data member: type or color of the tile.
 - (b) Methods: `getType()`, `setType()`.
2. Create a `Board` class to represent the game grid.

- (a) Initialize the grid with random tiles.
 - (b) Display the current state of the grid in a readable format.
 - (c) Detect and remove matches of three or more tiles in a row or column.
 - (d) Drop new tiles from the top to fill empty spaces.
 - (e) Include a method to check if possible moves exist.
3. Create a **Game** class to manage gameplay.
 - (a) Display options for the user: **Play** or **Solution**.
 - (b) Allow the user to swap adjacent tiles using input commands.
 - (c) Keep track of the player's score.
 - (d) Allow the user to quit and confirm before exiting.
 - (e) Optionally, allow the user to reset the board or play multiple rounds.
 4. Ensure proper encapsulation of tile and board operations within the respective classes.

Suggested Class Structure

1. **Tile** Class:

- Data member: `int type` or `char color`
- Methods: `getType()`, `setType()`

2. **Board** Class:

- Data member: 2D array of **Tile** objects
- Methods: `initializeBoard()`, `displayBoard()`, `detectMatches()`, `removeMatches()`, `dropTiles()`, `hasPossibleMoves()`

3. **Game** Class:

- Data member: **Board** object, user choice, score
- Methods: `playGame()`, `showSolution()`, `processSwap(int row1, int col1, int row2, int col2)`, `confirmQuit()`, `resetBoard()`

Reference

For more details about Match-3 games and mechanics, visit: https://en.wikipedia.org/wiki/Tile-matching_video_game