REPUBLIC OF THE PHILIPPINES

BICOL UNIVERSITY

BICOL UNIVERSITY POLANGUI

PARTNER NAME: Curt Justin N. Reodique/Faith Ann Sañado

SUBJECT: Data Structures and Algorithms

COURSE/YEAR: BSIS - 2

PROF: Ms. Kristine Botin

SECTION: 2A

Title: Superhero Mission Scheduler Using Min-Heap

Theme: The program manages superhero missions by prioritizing tasks using a Min-Heap. The priority of each mission is determined by its urgency (lower numerical value indicates higher priority). The heap data structure efficiently organizes and processes tasks to ensure the most critical missions are addressed first. This program builds skills in heap implementation, task prioritization, and efficient data management.

Learning Objectives:

- **Heap Data Structures:** Understand Min-Heaps and their use as priority queues.
- **Heap Operations:** Learn heapify-up (insertion), heapify-down (deletion), and heapify an array.
- **Priority Management:** Dynamically manage tasks based on priority using heaps.
- **Working with Pairs:** Use std::pair to associate tasks with priorities.
- **Algorithmic Thinking:** Analyze and optimize heap operations for time complexity.
- **Simulation and Problem Solving:** Simulate real-world scheduling problems with heaps.
- **Debugging and Visualization:** Visualize heap states to ensure operation correctness.

Tasks:

1. Inserting Missions into the Min-Heap
   - Adds new missions to the heap with a given priority.
   - Ensures the heap property (smallest priority at the root) is maintained using the heapify-up operation.

2. Deleting the Highest Priority Mission
   - Removes the mission with the highest priority (root of the Min-Heap).
   - Restores the heap property using the heapify-down operation.

3. Heapifying an Array of Missions into a Min-Heap
   - Converts an unsorted array of missions into a Min-Heap.
   - Uses the heapify-down operation starting from non-leaf nodes to build the heap efficiently.

4. Displaying Missions
   - Prints the current state of missions in the heap for visualization.
   - Allows users to see the order of mission priorities at different stages.

INSTRUCTIONS:

Input

- The inputs are embedded in the code as predefined data:
- Task 1: Insert missions into the Min-Heap:

  "Stop the bank robbery" with priority 1.

  "Defuse the bomb" with priority 2.

  "Rescue the cat" with priority 5.

  "Apprehend the villain" with priority 3.

- Task 2: Delete the mission with the highest priority (the root).
- Task 3: Transform the following unordered array into a Min-Heap:

  "Save the world": priority 1.

  "Find the missing dog": priority 10.

  "Stop the asteroid": priority 2.

  "Deliver pizza": priority 9.

Output

- Task 1: Insert Missions
  Adding mission 'Stop the bank robbery' with priority 1 to the Min-Heap!
  Adding mission 'Defuse the bomb' with priority 2 to the Min-Heap!
  Adding mission 'Rescue the cat' with priority 5 to the Min-Heap!
  Adding mission 'Apprehend the villain' with priority 3 to the Min-Heap!
  Current Mission Priorities (Min-Heap):
  Stop the bank robbery: 1
  Defuse the bomb: 2
  Rescue the cat: 5
  Apprehend the villain: 3

- Task 2: Delete the Highest Priority Mission
  Deleting mission 'Stop the bank robbery' (highest priority) from the Min-Heap!
  Current Mission Priorities (Min-Heap):
  Defuse the bomb: 2
  Apprehend the villain: 3
  Rescue the cat: 5
- Task 3: Heapify an Array
  Initial Missions (Unordered):
  Save the world: 1
  Find the missing dog: 10
  Stop the asteroid: 2
  Deliver pizza: 9

  Missions after Heapify (Min-Heap):
  Save the world: 1
  Deliver pizza: 9
  Stop the asteroid: 2
  Find the missing dog: 10

Code Analysis

Flow Summary:

Initialize an empty heap (mission Heap).

Insert missions into the heap with specific priorities.

After each insertion, heapify-up ensures the Min-Heap property is maintained.

Delete the mission with the highest priority (root of the heap).

After deletion, heapify-down ensures the Min-Heap property is restored.

Heapify an unordered array of missions to form a valid Min-Heap.

The array is converted using the heapify-down method.

Print the heap after each modification (insertion, deletion, heapify).

VISUAL FLOW:

Start

```
|
├─> Initialize empty Min-Heap (missionHeap)
|
├─> Task 1: Insert missions
|   ├─> Insert mission: “Stop the bank robbery” (Priority 1)
|   ├─> Insert mission: “Defuse the bomb” (Priority 2)
|   ├─> Insert mission: “Rescue the cat” (Priority 5)
|   ├─> Insert mission: “Apprehend the villain” (Priority 3)
|
├─> Print the current state of the Min-Heap
|
├─> Task 2: Delete the highest priority mission (root)
|   ├─> Delete root: “Stop the bank robbery” (Priority 1)
|
├─> Print the current state of the Min-Heap after deletion
|
├─> Task 3: Heapify an unordered array of missions
|   ├─> Input unordered array of missions
|   ├─> Heapify the array into a Min-Heap
|
├─> Print the Min-Heap after heapification
|
End
```

SUMMARY:

The c++ program implements the “Superhero Mission Scheduler Using Min-Heap” provides a practical application of the Min-Heap data structure, where users can manage superhero missions efficiently based on priority. The game introduces fundamental heap operations like insertion, deletion, and heapification, making it an educational tool for understanding how heaps work in programming.