Republic of the Philippines

Bicol University

Bicol University Polangui

Computer Studies Department

Polangui Albay

Shanley R. Resentes

Penelope R. Sablayan

BSIS-2A

DATA STUCTURE AND ALGORITHM

Design Your Heap Challenge

Title: E-Sports Tournament
Theme: Sort teams or players by ranking or skill level

Tasks:

1. Insert Players into the tournament

- Add players to the Max-Heap by providing their skill levels.
- Observe how the heap reorganizes itself to maintain the Max-Heap property, with the highest-skilled player always at the top

2. View weakest players

- Convert the current Max-Heap to a Min-Heap to view the weakest players in the tournament.
- Use this feature to plan for eliminations or to assign training priorities to lower-ranked players.

3. Update rankings with a new batch of players

- Enter a list of new players with their skill levels
- Observe how the program reorganizes the rankings into a Max-Heap (with the strongest players at the top)
- Compare the new rankings with the previous rankings for insights.

4. Test exit functionality

- Test the exit functionality by selecting option four(4)
- Confirm that the progra, gracefully exits with a goodbye message.

5. Handle duplicate skill levels

- Add multiple players with the same skill level and observe how the heap handles duplicates values.

6. Manage empty heaps

- Try viewing the weakest players (option 2) or updating rankings (option 3) when the heap is empty.
- Observe if the program handles this gracefully without crashing

7. Test large inputs

- Add many players to the tournament to test the program's efficiency with large heaps.
- Observe how quickly the heap reorganizes for insertions and conversions.

8. Simulate a full tournament

- Add a set of players to the tournament
- Convert the heap to a Min-Heap to identify weaker players
- Update the heap rankings with a new list of players to simulate the next around of the tournament.

9. Experiment with edge cases

- Add skill levels outside the valid range (e.g. negative numbers, numbers above 100)
- Observe how the program responds or modify it to handle invalid inputs gracefully.

10. Add enhancements

- Modify the program to include: Player names: Associate a name with each skill level for better identification.
- Tournament phases: allow players to be removed from the heap (e.g. disqualify weakest players)
- Leaderboard: Display the top 3 players in the Max-Heap

INPUT: Adding a player in E-sports tournament ranking system

OUTPUT: Player with skill level has joined the tournament

SAMPLE CODE:

```
        #include <iostream>

#include <vector>
```

```
#include <algorithm>

using namespace std;


// Function to maintain Max-Heap property after insertion

void heapifyUp(vector<int> &heap, int index) {

   int parent = (index - 1) / 2;

   if (index > 0 && heap[index] > heap[parent]) {

      swap(heap[index], heap[parent]);

      heapifyUp(heap, parent);

   }

}


// Insert a new player into the Max-Heap

void insertPlayer(vector<int> &heap, int skillLevel) {

   heap.push_back(skillLevel);

   cout << "Player with skill level " << skillLevel << " has joined the tournament!\n";

   heapifyUp(heap, heap.size() - 1);

   cout << "Current Max-Heap (Top players): ";

   for (int val : heap) cout << val << " ";

   cout << endl;

}
```

```
// Function to maintain Min-Heap property

void heapifyDown(vector<int> &heap, int index, int size) {

   int left = 2 * index + 1;

   int right = 2 * index + 2;

   int smallest = index;


   if (left < size && heap[left] < heap[smallest]) smallest = left;

   if (right < size && heap[right] < heap[smallest]) smallest = right;


   if (smallest != index) {

      swap(heap[index], heap[smallest]);

      heapifyDown(heap, smallest, size);

   }

}


// Convert Max-Heap to Min-Heap

void convertToMinHeap(vector<int> &heap) {

   for (int i = (heap.size() / 2) - 1; i >= 0; i--) {

      heapifyDown(heap, i, heap.size());

   }

   cout << "Converted Max-Heap to Min-Heap (Weakest players first): ";

   for (int val : heap) cout << val << " ";
```

```
    cout << endl;
}


// Function to heapify a Max-Heap from an unordered array
void heapifyToMaxHeap(vector<int> &arr) {
    for (int i = (arr.size() / 2) - 1; i >= 0; i--) {
        heapifyDown(arr, i, arr.size());
    }
    cout << "Heapified Max-Heap (Rankings updated): ";
    for (int val : arr) cout << val << " ";
    cout << endl;
}


int main() {
    vector<int> heap;
    int choice, skillLevel;


    cout << "=== E-Sports Tournament Ranking System ===\n";
    while (true) {
        cout << "\n1. Add Player\n2. Show Weakest Players (Convert to Min-Heap)\n";
        cout << "3. Update Rankings from New Players\n4. Exit\n";
        cout << "Enter your choice: ";
```

```
cin >> choice;


switch (choice) {
    case 1:
        cout << "Enter player's skill level (1-100): ";
        cin >> skillLevel;
        insertPlayer(heap, skillLevel);
        break;


    case 2:
        convertToMinHeap(heap);
        break;


    case 3: {
        vector<int> newPlayers;
        int numPlayers;
        cout << "Enter the number of new players: ";
        cin >> numPlayers;
        cout << "Enter the skill levels of the new players:\n";
        for (int i = 0; i < numPlayers; i++) {
            cin >> skillLevel;
            newPlayers.push_back(skillLevel);
```

```
                }
                heapifyToMaxHeap(newPlayers);
                break;
            }

            case 4:
                cout << "Exiting Tournament Ranking System. Goodbye!\n";
                return 0;

            default:
                cout << "Invalid choice. Please try again.\n";
        }
    }
}
```