Aloysious Irish O. Portugal

Kharl R. Repato **BSCS 2 A**

Heap Quest: The Min-Max Mastery Challenge Welcome to the magical land of HeapQuest! You are the Keeper of Balance, tasked with maintaining harmony in the Heaps of Power. Solve these tasks to protect the realm.

Task 1: The Great Heap Insertion

Scenario: You've received a shipment of enchanted artifacts with varying power levels. Insert them into a Max-Heap to ensure the most powerful artifact remains on top.

Instructions:

Write a function insertToHeap to insert elements into a Max-Heap. Maintain the heap property after each insertion. Display the heap array after each insertion.

Example Input: Insert artifacts with power levels [10, 25, 5, 30, 15].

Expected Output:

Inserting artifact with power 10: [10]

Inserting artifact with power 25: [25, 10]

Inserting artifact with power 5: [25, 10, 5]

Inserting artifact with power 30: [30, 25, 5, 10]

Inserting artifact with power 15: [30, 25, 5, 10, 15]

Code Template:

```
#include <iostream>

#include <vector>

using namespace std;

// Function to heapify upwards

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

while (index > 0) {

int parent = (index - 1) / 2;

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

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

index = parent;
```

```
} else {
break;
}
 }
 }
// Function to insert into the Max-Heap
void insertToHeap(vector<int>& heap, int value) {
heap.push_back(value);
// Add the new value
heapifyUp(heap, heap.size() - 1);
// Restore heap property
cout << "Current Heap: ";
for (int num : heap)
cout << num << " ";
cout << endl;
}
int main() {
vector<int> maxHeap;
vector<int> values = {10, 25, 5, 30, 15};
for (int value : values) {
cout << "Inserting artifact with power " << value << ": "; insertToHeap(maxHeap, value);
 }
return 0;
}
```

Task 2: The Root's Demise Scenario: A powerful artifact at the root of your heap is damaged and must be removed. Your task is to delete the root while maintaining the Max-Heap property.

Instructions:

Write a function deleteRoot to remove the root (maximum value). Restore the heap property after deletion using heapifyDown. Display the heap array after the root is removed.

Example Input: Start with the heap [30, 25, 5, 10, 15].

Expected Output:

Removing root 30: [25, 15, 5, 10]

Removing root 25: [15, 10, 5] Code Template:

```
// Function to heapify downwards
void heapifyDown(vector<int>& heap, int index) {
int size = heap.size();
while (true) {
int left = 2 * index + 1;
int right = 2 * index + 2;
int largest = index;
if (left < size && heap[left] > heap[largest]) largest = left;
if (right < size && heap[right] > heap[largest]) largest = right;
if (largest != index) {
swap(heap[index], heap[largest]);
index = largest;
 } else {
break;
}
 }
}
// Function to delete the root from the Max-Heap
void deleteRoot(vector<int>& heap) {
if (heap.empty()) return;
cout << "Removing root " << heap[0] << ": "; heap[0] = heap.back();
// Replace root with last element
heap.pop_back();
// Remove last element
heapifyDown(heap, 0);
```

```
// Restore heap property

for (int num : heap)

cout << num << " ";

cout << endl;

}

int main() {

vector<int> maxHeap = {30, 25, 5, 10, 15};

while (!maxHeap.empty()) {

deleteRoot(maxHeap);

}

 return 0;
```

} **Task 3:** Heapify the Array Scenario: A disordered collection of enchanted gems has arrived. Your task is to convert the array into a Max-Heap using the Heapify process.

Instructions: Implement a function buildMaxHeap to heapify an array into a Max-Heap. Use the heapifyDown approach starting from the last non-leaf node. Display the array before and after heapification.

Example Input: [5, 15, 10, 25, 30]

Expected Output:

Before heapification: [5, 15, 10, 25, 30]

After heapification: [30, 25, 10, 5, 15]

Code Template:

```
void buildMaxHeap(vector<int>& heap) {

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

heapifyDown(heap, i);

}

}

int main() {

vector<int> gems = {5, 15, 10, 25, 30};

cout << "Before heapification: ";

for (int num : gems)
```

```
cout << num << " ";

cout << endl;

buildMaxHeap(gems);

cout << "After heapification: ";

for (int num : gems)

cout << num << " ";

cout << endl;

return 0;

}
```

Step 5: Make It Fun! Add themed print statements like: “Inserting artifact with power X into the Heap of Power!” “Removing the root artifact with power X to maintain harmony!” Visualize the heap as a tree structure using indentation or levels for a magical effect. These tasks are designed to build familiarity with heap operations in an engaging and step-by-step manner