MERGE SORT 2A

Joey Welvaadt

HOGESCHOOL UTRECHT 1734098

1. Code

```cpp
#include <iostream>
#include <vector>
#include "include/mergesort.hpp"

using namespace std;
using namespace mergesort;

int main() {
    vector<int> vec = {5, 4, 3, 2, 1};
    merge_sort(vec, 0, vec.size() - 1);

    cout << "Unsorted vector: ";
    for (int i = 0; i < vec.size(); i++) {
        cout << vec[i] << " ";
    }
    cout << endl;


    cout << "Sorted vector: ";
    for (int i = 0; i < vec.size(); i++) {
        cout << vec[i] << " ";
    }
    cout << endl;

    return 0;
}
```

```cpp
#include "include/mergesort.hpp"

// Source: https://www.mygreatlearning.com/blog/merge-sort/

namespace mergesort {

    void merge_sort(vector<int>& vec, int left, int right) {
        // When the left index is bigger than the right index, we are working with an
        // invalid range and we return. Closing this path.
        if (left >= right) {
            return;
        }

        // Get the middle index, take for example a vector of 5 elements:
        // Left would be 0 and right would be 4. The sum would be left + (right - left) / 2
        // 0 + (4 - 0) / 2 = 2
        // Getting the new right and left would be as simple as: left & middle, and middle + 1 & right
        int middle = left + (right - left) / 2;

        // Sort the left and right halves
        merge_sort(vec, left, middle);
        merge_sort(vec, middle + 1, right);

        // Merge the two halves
        merge(vec, left, middle, right);
    }

    void merge(vector<int>& vec, int left, int middle, int right) {
        // Get the length of the left and right halves, below follows and example given a vector of 5 elements:
        // len1 = middle - left + 1, example: 2 - 0 + 1 = 3
        int len1 = middle - left + 1;
        // len2 = right - middle, example: 4 - 2 = 2
        int len2 = right - middle;

        // Create two temporary vectors to store the left and right halves
        vector<int> left_vec(len1);
        vector<int> right_vec(len2);

        // Fill the temporary vectors with the left and right halves
        for (int i = 0; i < len1; i++) {
            left_vec[i] = vec[left + i];
        }
        for (int j = 0; j < len2; j++) {
            right_vec[j] = vec[middle + 1 + j];
        }

        // Initialize the indices for the left and right halves
        int l_index = 0, r_index = 0, vec_index = left;

        // Merge the two halves
        while (l_index < len1 && r_index < len2) {
            if (left_vec[l_index] <= right_vec[r_index]) {
                vec[vec_index] = left_vec[l_index];
                l_index++;
            } else {
                vec[vec_index] = right_vec[r_index];
                r_index++;
            }
            vec_index++;
        }

        // Copy the remaining elements of left_vec, if there are any
        while (l_index < len1) {
            vec[vec_index] = left_vec[l_index];
            l_index++;
            vec_index++;
        }

        // Copy the remaining elements of right_vec, if there are any
        while (r_index < len2) {
            vec[vec_index] = right_vec[r_index];
            r_index++;
            vec_index++;
        }
    }
}
```

2. Output

```
C:\Users\joeyw\Projects\School\Jaar 2\
Projects\School\Jaar 2\Blok C\HPP\Herk
Unsorted vector: 1 2 3 4 5
Sorted vector: 1 2 3 4 5
```