WWC_DAY-2

Name: Siddharth Lal

UID: 22BCS13410

Problem_1

main.cpp

```cpp
#include <iostream>
#include <vector>

using namespace std;

int singleNumber(vector<int>& nums) {
    int result = 0;
    for (int num : nums) {
        result ^= num;
    }
    return result;
}

int main() {
    vector<int> nums1 = {2, 2, 1};
    cout << "Output: " << singleNumber(nums1) << endl;

    vector<int> nums2 = {4, 1, 2, 1, 2};
    cout << "Output: " << singleNumber(nums2) << endl;

    return 0;
}
```

input

```
Output: 1
Output: 4
```

Problem_2

main.cpp

```cpp
#include <iostream>
#include <vector>
using namespace std;

vector<vector<int>> generatePascalsTriangle(int numRows) {
    vector<vector<int>> triangle;

    for (int i = 0; i < numRows; ++i) {
        vector<int> row(i + 1, 1);

        for (int j = 1; j < i; ++j) {
            row[j] = triangle[i - 1][j - 1] + triangle[i - 1][j];
        }

        triangle.push_back(row);
    }

    return triangle;
}

int main() {
    int numRows;
    cout<<"Enter the number of rows: ";
    cin >> numRows;
```

```cpp
int main() {
    int numRows;
    cout<<"Enter the number of rows: ";
    cin >> numRows;

    vector<vector<int>> triangle = generatePascalsTriangle(numRows);

    for (const auto& row : triangle) {
        for (int num : row) {
            cout << num << " ";
        }
        cout << endl;
    }

    return 0;
}
```

input

```
Enter the number of rows: 5
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

Problem_3

main.cpp

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int maxArea(const vector<int>& height) {
    int left = 0, right = height.size() - 1;
    int maxWater = 0;

    while (left < right) {
        int width = right - left;
        int minHeight = min(height[left], height[right]);
        maxWater = max(maxWater, width * minHeight);

        if (height[left] < height[right]) {
            left++;
        } else {
            right--;
        }
    }

    return maxWater;
}
```

input

```
Maximum water: 49
```

Problem_4

```cpp
#include <iostream>
#include <vector>
using namespace std;

int maxHappyGroups(int batchSize, vector<int>& groups) {
    vector<int> remainderCount(batchSize, 0);

    for (int group : groups) {
        remainderCount[group % batchSize]++;
    }

    int happyGroups = remainderCount[0];

    for (int i = 1; i < batchSize; ++i) {
        int complement = batchSize - i;
        if (remainderCount[i] > 0) {
            if (i == complement) {
                happyGroups += remainderCount[i] / 2;
                remainderCount[i] %= 2;
            } else {
                int pairs = min(remainderCount[i], remainderCount[complement]);
                happyGroups += pairs;
                remainderCount[i] -= pairs;
                remainderCount[complement] -= pairs;
            }
        }
    }

    int remaining = 0;
    for (int i = 1; i < batchSize; ++i) {
        remaining += remainderCount[i] * i;
    }

    return happyGroups + (remaining / batchSize);
}

int main() {
    vector<int> groups1 = {1, 2, 3, 4, 5, 6};
    cout << "Example 1 Output: " << maxHappyGroups(3, groups1) << endl;

    vector<int> groups2 = {1, 3, 2, 5, 2, 2, 1, 6};
    cout << "Example 2 Output: " << maxHappyGroups(4, groups2) << endl;

    return 0;
}
```

main.cpp

input

```
Example 1 Output: 4
Example 2 Output: 3
```

Problem_5

```
        return false;
    };

    return backtrack(0);
}

int minimumTimeRequired(vector<int>& jobs, int k) {
    sort(jobs.rbegin(), jobs.rend());

    int left = jobs[0], right = 0;
    for (int job : jobs)
        right += job;

    while (left < right) {
        int mid = left + (right - left) / 2;
        if (canAssignJobs(jobs, k, mid)) {
            right = mid;
        } else {
            left = mid + 1;
        }
    }

    return left;
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

bool canAssignJobs(const vector<int>& jobs, int k, int maxTime) {
    vector<int> workers(k, 0);

    function<bool(int)> backtrack = [&](int i) -> bool {
        if (i == jobs.size())
            return true;

        for (int j = 0; j < k; ++j) {
            if (workers[j] + jobs[i] <= maxTime) {
                workers[j] += jobs[i];
                if (backtrack(i + 1))
                    return true;
                workers[j] -= jobs[i];
            }
            if (workers[j] == 0)
                break;
        }
        return false;
    };
```

```
            left = mid + 1;
        }
    }

    return left;
}

int main() {
    vector<int> jobs1 = {3, 2, 3};
    int k1 = 3;
    cout << "Output: " << minimumTimeRequired(jobs1, k1) << endl;

    vector<int> jobs2 = {1, 2, 4, 7, 8};
    int k2 = 2;
    cout << "Output: " << minimumTimeRequired(jobs2, k2) << endl;

    return 0;
}
```

input

```
Output: 3
Output: 11
```