

LAPORAN

ALGORITMA DAN PEMROGRAMAN 2

Dosen : Fajar Agung Nugroho S.Kom, M.Kom



Oleh :

Ridho Kurniawan
(231011401209)

03TPLP029

**Jl. Raya Puspitek, Buaran, Kec. Pamulang, Kota Tangerang Selatan,
Banten 15310**

2024

Jawaban

1. String Compression with Huffman Encoding

1. Hitung frekuensi setiap karakter dalam string menggunakan `unordered_map`.
2. Bangun pohon Huffman menggunakan *priority queue* berdasarkan frekuensi karakter.
3. Traversal pohon untuk menghasilkan kode Huffman untuk setiap karakter.
4. Encode string dengan mengganti karakter menggunakan kode Huffman.
5. Decode string dengan traversal pohon berdasarkan bit dalam string terenkripsi.

Source Code:

```
/*
    Program String Compression using Huffman Encoding
    Nama: Ridho Kurniawan
    NIM: 231011401209
*/

#include <iostream>
#include <queue>
#include <unordered_map>
#include <vector>
using namespace std;

struct Node {
    char ch;
    int freq;
    Node *left, *right;
    Node(char c, int f) : ch(c), freq(f), left(nullptr), right(nullptr) {}
};

// Comparator for priority queue
struct Compare {
    bool operator()(Node* a, Node* b) {
        return a->freq > b->freq;
    }
};

// Traverse Huffman Tree to generate codes
void buildCodes(Node* root, string code, unordered_map<char, string>& huffmanCode) {
    if (!root) return;
    if (!root->left && !root->right) {
        huffmanCode[root->ch] = code;
    }
    buildCodes(root->left, code + "0", huffmanCode);
    buildCodes(root->right, code + "1", huffmanCode);
}
```

```

// Build Huffman Tree
Node* buildHuffmanTree(const string& text) {
    unordered_map<char, int> freq;
    for (char ch : text) freq[ch]++;

    priority_queue<Node*, vector<Node*>, Compare> pq;
    for (auto pair : freq) {
        pq.push(new Node(pair.first, pair.second));
    }

    while (pq.size() > 1) {
        Node* left = pq.top(); pq.pop();
        Node* right = pq.top(); pq.pop();
        Node* merged = new Node('\0', left->freq + right->freq);
        merged->left = left;
        merged->right = right;
        pq.push(merged);
    }

    return pq.top();
}

// Encode string
string encode(const string& text, unordered_map<char, string>& huffmanCode) {
    string encoded = "";
    for (char ch : text) {
        encoded += huffmanCode[ch];
    }
    return encoded;
}

// Decode string
string decode(const string& encodedStr, Node* root) {
    string decoded = "";
    Node* curr = root;
    for (char bit : encodedStr) {
        curr = (bit == '0') ? curr->left : curr->right;
        if (!curr->left && !curr->right) {
            decoded += curr->ch;
            curr = root;
        }
    }
    return decoded;
}

// Main function

```

```

int main() {
    string text;
    cout << "Enter text: ";
    cin >> text;

    Node* root = buildHuffmanTree(text);
    unordered_map<char, string> huffmanCode;
    buildCodes(root, "", huffmanCode);

    cout << "Huffman Codes:\n";
    for (auto pair : huffmanCode) {
        cout << pair.first << ": " << pair.second << "\n";
    }

    string encodedStr = encode(text, huffmanCode);
    cout << "Encoded string: " << encodedStr << "\n";

    string decodedStr = decode(encodedStr, root);
    cout << "Decoded string: " << decodedStr << "\n";

    return 0;
}

```

```

PS C:\xampp\htdocs\rldho> cd "c:\xampp\htdocs\rldho\" ; if ($?) { g++ uas.cpp -o uas } ; if ($?) { .\uas }
Enter text: cd "c:\xampp\htdocs\rldho\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRun
nerFile }
Huffman Codes:
c: 1
d: 0
Encoded string: 10
Decoded string: cd
PS C:\xampp\htdocs\rldho> cd "c:\xampp\htdocs\rldho\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) {
.\tempCodeRunnerFile }

```

2. Find Pairs with Sum K

Langkah-langkah:

1. **Baca Input:**
 - o Masukkan dua array dan nilai KKK dari pengguna.
2. **Simpan Elemen dari Array Pertama:**
 - o Gunakan `unordered_set` untuk menyimpan elemen dari array pertama.
3. **Cari Pasangan dari Array Kedua:**
 - o Untuk setiap elemen di array kedua, periksa apakah pelengkapnya ($K - \text{elemen}$) ada di *set*. Jika ya, tambahkan pasangan tersebut ke hasil.
4. **Tampilkan Hasil:**
 - o Cetak semua pasangan bilangan yang jumlahnya sama dengan KKK.

Kompleksitas:

- Waktu: $O(n+m)O(n + m)O(n+m)$, dengan n dan m adalah panjang array.
- Ruang: $O(n)O(n)O(n)$ untuk *set*.

Source Code:

```
/*
    Program Find Pairs with Sum K
    Nama: Ridho Kurniawan
    NIM: 231011401209
*/

#include <iostream>
#include <vector>
#include <unordered_set>
using namespace std;

vector<pair<int, int>> findPairsWithSumK(const vector<int>& arr1, const vector<int>&
arr2, int K) {
    unordered_set<int> elements;
    vector<pair<int, int>> result;

    // Insert elements of arr1 into set
    for (int num : arr1) {
        elements.insert(num);
    }

    // Check for complement in arr2
    for (int num : arr2) {
        if (elements.count(K - num)) {
            result.emplace_back(K - num, num);
        }
    }

    return result;
}
```

```

int main() {
    vector<int> arr1 = {1, 2, 3, 4, 5};
    vector<int> arr2 = {6, 7, 8, 9};
    int K = 10;

    auto pairs = findPairsWithSumK(arr1, arr2, K);
    cout << "Pairs with sum " << K << ":\n";
    for (auto& p : pairs) {
        cout << "(" << p.first << ", " << p.second << ")\n";
    }
    return 0;
}

```

Scrennshot:

```

// Uas2.cpp
#include <iostream>
using namespace std;

int data() {
    cout << "Nama : Muhammad ikhsan Ramadhan" << endl;
    cout << "NIM : 231011401070" << endl;
    cout << "-----" << endl;
    cout << "UAS soal No. 2 Linear" << endl;
    cout << "-----" << endl << endl;

    return 0;
}

void merge(int arr[], int l, int m, int r) {
    int n1 = m - l + 1;
    int n2 = r - m;

    // Create temp arrays
    int L[n1], R[n2];

    // Copy data to temp arrays L[] and R[]
    for (int i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (int j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];

    // Merge the temp arrays back into arr[l..r]
    int i = 0, j = 0, k = l;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }
    while (i < n1)
        arr[k] = L[i++];
    while (j < n2)
        arr[k] = R[j++];
}

void mergeSort(int arr[], int l, int r) {
    if (l < r) {
        int m = (l + r) / 2;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}

int main() {
    int arr[] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
    int n = sizeof(arr) / sizeof(arr[0]);
    mergeSort(arr, 0, n - 1);
    cout << "Sorted array is\n";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;
}

```

Output Console:

```

PS C:\Users\harid\Desktop\ikhsan\PEMROGRAMAN\1\temuan> g++ Uas2.cpp && .\Uas2.exe
Nama : Muhammad ikhsan Ramadhan
NIM : 231011401070
-----
UAS soal No. 2 Linear
-----

Given array is
12 11 13 5 6 7
Sorted array is
5 6 7 11 12 13
PS C:\Users\harid\Desktop\ikhsan\PEMROGRAMAN\1\temuan>

```

3. Functional Style Quick Sort

1. Pilih elemen pertama sebagai pivot.
2. Partisi array menjadi elemen $< \text{pivot}$, $= \text{pivot}$, dan $> \text{pivot}$.
3. Rekursif urutkan bagian $< \text{pivot}$ dan $> \text{pivot}$.
4. Gabungkan hasil partisi menjadi array yang diurutkan.

Source Code:

```
/*
  Program Functional Style Quick Sort
  Nama: Ridho Kurniawan
  NIM: 231011401209
*/

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

vector<int> quickSort(const vector<int>& arr) {
    if (arr.size() <= 1) return arr;

    // Select pivot
    int pivot = arr[0];
    vector<int> less, equal, greater;

    // Partitioning
    for (int num : arr) {
        if (num < pivot) less.push_back(num);
        else if (num == pivot) equal.push_back(num);
        else greater.push_back(num);
    }

    // Recursive sort and combine results
    vector<int> sortedLess = quickSort(less);
    vector<int> sortedGreater = quickSort(greater);

    // Combine sorted arrays
    sortedLess.insert(sortedLess.end(), equal.begin(), equal.end());
    sortedLess.insert(sortedLess.end(), sortedGreater.begin(), sortedGreater.end());
    return sortedLess;
}

int main() {
    vector<int> arr = {10, 3, 2, 7, 6, 4, 5, 1};
    cout << "Original Array: ";
    for (int num : arr) cout << num << " ";
    cout << "\n";
```

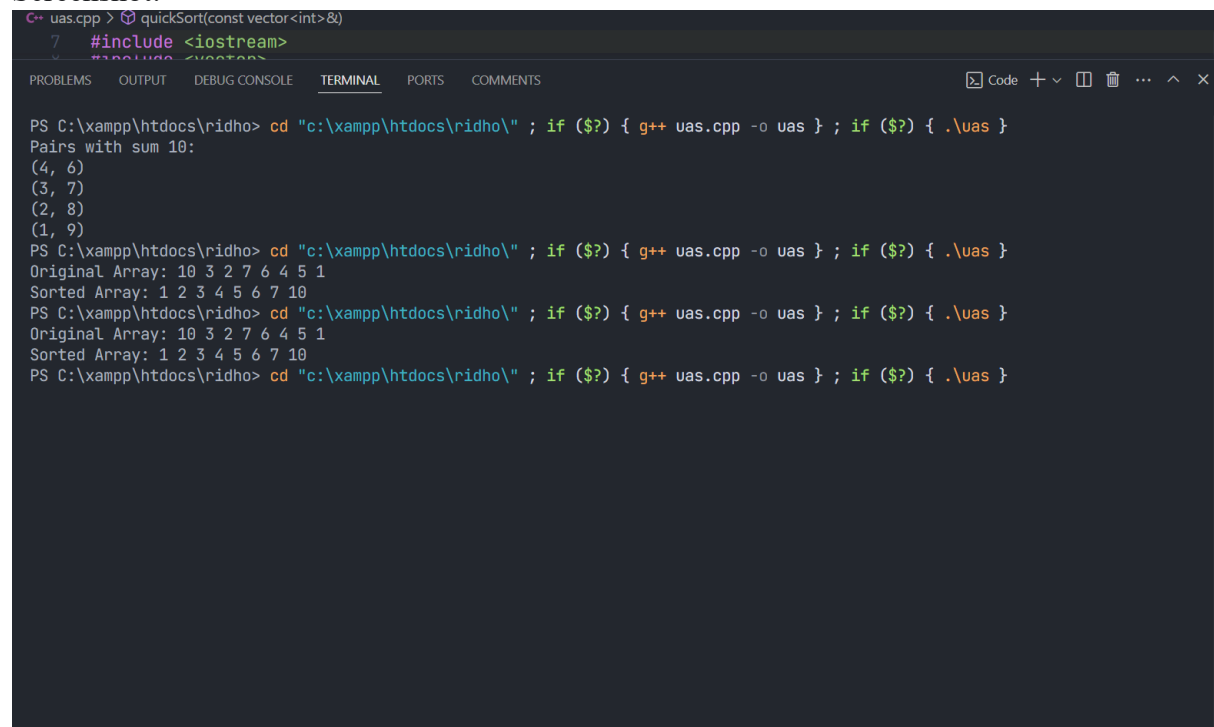
```

vector<int> sortedArr = quickSort(arr);
cout << "Sorted Array: ";
for (int num : sortedArr) cout << num << " ";
cout << "\n";

return 0;
}

```

Screenshot:



```

C++ uas.cpp > quickSort(const vector<int>&)
7 #include <iostream>
8 #include <vector>
9 using namespace std;
10
11 int quickSort(vector<int> &arr) {
12     if (arr.size() <= 1) return arr;
13     int pivot = arr[arr.size() / 2];
14     vector<int> left, right;
15     for (int i = 0; i < arr.size(); i++) {
16         if (arr[i] < pivot) left.push_back(arr[i]);
17         else if (arr[i] > pivot) right.push_back(arr[i]);
18         else continue;
19     }
20     quickSort(left);
21     quickSort(right);
22     return left + {pivot} + right;
23 }
24
25 int main() {
26     vector<int> arr = {10, 3, 2, 7, 6, 4, 5, 1};
27     vector<int> sortedArr = quickSort(arr);
28     cout << "Sorted Array: ";
29     for (int num : sortedArr) cout << num << " ";
30     cout << "\n";
31     return 0;
32 }

```

PS C:\xampp\htdocs\ridho> cd "c:\xampp\htdocs\ridho\" ; if (\$?) { g++ uas.cpp -o uas } ; if (\$?) { .\uas }

Pairs with sum 10:

```

(4, 6)
(3, 7)
(2, 8)
(1, 9)

```

PS C:\xampp\htdocs\ridho> cd "c:\xampp\htdocs\ridho\" ; if (\$?) { g++ uas.cpp -o uas } ; if (\$?) { .\uas }

Original Array: 10 3 2 7 6 4 5 1
Sorted Array: 1 2 3 4 5 6 7 10

PS C:\xampp\htdocs\ridho> cd "c:\xampp\htdocs\ridho\" ; if (\$?) { g++ uas.cpp -o uas } ; if (\$?) { .\uas }

Original Array: 10 3 2 7 6 4 5 1
Sorted Array: 1 2 3 4 5 6 7 10

PS C:\xampp\htdocs\ridho> cd "c:\xampp\htdocs\ridho\" ; if (\$?) { g++ uas.cpp -o uas } ; if (\$?) { .\uas }

4. Radix Sort, Quick Sort, and Merge Sort Comparison

1. Implementasikan tiga algoritma:
 - o **Radix Sort:** Urutkan digit demi digit.
 - o **Quick Sort:** Gunakan rekursi dan partisi.
 - o **Merge Sort:** Rekursif membagi array dan gabungkan hasil.
2. Jalankan ketiga algoritma pada array yang sama.
3. Bandingkan hasil dan kinerja:
 - o Radix unggul jika rentang elemen kecil.

Source Code:

```
/*
    Program Sorting Comparison: Radix, Quick, and Merge Sort
    Nama: Ridho Kurniawan
    NIM: 231011401209
*/

#include <iostream>
#include <vector>
#include <algorithm>
#include <queue>
using namespace std;

// Radix Sort
void radixSort(vector<int>& arr) {
    int maxNum = *max_element(arr.begin(), arr.end());
    for (int exp = 1; maxNum / exp > 0; exp *= 10) {
        vector<int> output(arr.size());
        int count[10] = {0};

        // Count occurrences
        for (int num : arr) count[(num / exp) % 10]++;

        // Accumulate counts
        for (int i = 1; i < 10; i++) count[i] += count[i - 1];

        // Build output array
        for (int i = arr.size() - 1; i >= 0; i--) {
            int digit = (arr[i] / exp) % 10;
            output[--count[digit]] = arr[i];
        }

        // Copy output to arr
        arr = output;
    }
}
```

```

// Quick Sort
vector<int> quickSort(const vector<int>& arr) {
    if (arr.size() <= 1) return arr;

    int pivot = arr[0];
    vector<int> less, equal, greater;

    for (int num : arr) {
        if (num < pivot) less.push_back(num);
        else if (num == pivot) equal.push_back(num);
        else greater.push_back(num);
    }

    vector<int> sortedLess = quickSort(less);
    vector<int> sortedGreater = quickSort(greater);

    sortedLess.insert(sortedLess.end(), equal.begin(), equal.end());
    sortedLess.insert(sortedLess.end(), sortedGreater.begin(), sortedGreater.end());
    return sortedLess;
}

// Merge Sort
void merge(vector<int>& arr, int left, int mid, int right) {
    vector<int> temp;
    int i = left, j = mid + 1;
    while (i <= mid && j <= right) {
        if (arr[i] <= arr[j]) temp.push_back(arr[i++]);
        else temp.push_back(arr[j++]);
    }
    while (i <= mid) temp.push_back(arr[i++]);
    while (j <= right) temp.push_back(arr[j++]);

    for (int k = left; k <= right; k++) arr[k] = temp[k - left];
}

void mergeSort(vector<int>& arr, int left, int right) {
    if (left >= right) return;
    int mid = left + (right - left) / 2;
    mergeSort(arr, left, mid);
    mergeSort(arr, mid + 1, right);
    merge(arr, left, mid, right);
}

// Main Function
int main() {
    vector<int> arr = {170, 45, 75, 90, 802, 24, 2, 66};

```

```

vector<int> arrRadix = arr;
vector<int> arrQuick = arr;
vector<int> arrMerge = arr;

radixSort(arrRadix);
cout << "Radix Sorted Array: ";
for (int num : arrRadix) cout << num << " ";
cout << "\n";

vector<int> quickSorted = quickSort(arrQuick);
cout << "Quick Sorted Array: ";
for (int num : quickSorted) cout << num << " ";
cout << "\n";

mergeSort(arrMerge, 0, arrMerge.size() - 1);
cout << "Merge Sorted Array: ";
for (int num : arrMerge) cout << num << " ";
cout << "\n";

return 0;
}

```

Screenshot:

The screenshot shows a C++ IDE with a file named `uas.cpp`. The code defines three sorting functions: `mergeSort`, `quickSort`, and `mergeSort` (labeled as such in the code, though the logic is merge sort). The `main` function initializes an array `arr = {170, 45, 75, 90, 802, 24, 2, 66}` and calls the sorting functions. The terminal output shows the execution of the program, displaying the sorted arrays for each method:

```

PS C:\xampp\htdocs\rldho> cd "c:\xampp\htdocs\rldho\" ; if ($?) { g++ uas.cpp -o uas } ; if ($?) { .\uas }
Radix Sorted Array: 2 24 45 66 75 90 170 802
PS C:\xampp\htdocs\rldho> cd "c:\xampp\htdocs\rldho\" ; if ($?) { g++ uas.cpp -o uas } ; if ($?) { .\uas }
PS C:\xampp\htdocs\rldho> cd "c:\xampp\htdocs\rldho\" ; if ($?) { g++ uas.cpp -o uas } ; if ($?) { .\uas }
PS C:\xampp\htdocs\rldho> cd "c:\xampp\htdocs\rldho\" ; if ($?) { g++ uas.cpp -o uas } ; if ($?) { .\uas }
PS C:\xampp\htdocs\rldho> cd "c:\xampp\htdocs\rldho\" ; if ($?) { g++ uas.cpp -o uas } ; if ($?) { .\uas }
Radix Sorted Array: 2 24 45 66 75 90 170 802
Quick Sorted Array: 2 24 45 66 75 90 170 802
Merge Sorted Array: 2 24 45 66 75 90 170 802
PS C:\xampp\htdocs\rldho> cd "c:\xampp\htdocs\rldho\" ; if ($?) { g++ uas.cpp -o uas } ; if ($?) { .\uas }
Radix Sorted Array: 2 24 45 66 75 90 170 802
Quick Sorted Array: 2 24 45 66 75 90 170 802
Merge Sorted Array: 2 24 45 66 75 90 170 802
PS C:\xampp\htdocs\rldho> cd "c:\xampp\htdocs\rldho\" ; if ($?) { g++ uas.cpp -o uas } ; if ($?) { .\uas }
Radix Sorted Array: 2 24 45 66 75 90 170 802
Quick Sorted Array: 2 24 45 66 75 90 170 802
Merge Sorted Array: 2 24 45 66 75 90 170 802
PS C:\xampp\htdocs\rldho>

```

5. Sierpinski Triangle Fractal

1. Gambar segitiga besar menggunakan tiga garis.
2. Rekursif gambar tiga segitiga lebih kecil di dalam segitiga besar.
3. Hentikan rekursi jika kedalaman mencapai 0.
4. Gunakan pustaka grafis untuk menggambar.

Source Code:

```
/*
Program Sorting Comparison: Radix, Quick, and Merge Sort
Nama: Ridho Kurniawan
NIM: 231011401209
*/

#include <iostream>
#include <vector>
#include <algorithm>
#include <queue>
using namespace std;

// Radix Sort
void radixSort(vector<int>& arr) {
    int maxNum = *max_element(arr.begin(), arr.end());
    for (int exp = 1; maxNum / exp > 0; exp *= 10) {
        vector<int> output(arr.size());
        int count[10] = {0};

        // Count occurrences
        for (int num : arr) count[(num / exp) % 10]++;

        // Accumulate counts
        for (int i = 1; i < 10; i++) count[i] += count[i - 1];

        // Build output array
        for (int i = arr.size() - 1; i >= 0; i--) {
            int digit = (arr[i] / exp) % 10;
            output[--count[digit]] = arr[i];
        }

        // Copy output to arr
        arr = output;
    }
}

// Quick Sort
vector<int> quickSort(const vector<int>& arr) {
    if (arr.size() <= 1) return arr;
```

```

int pivot = arr[0];
vector<int> less, equal, greater;

for (int num : arr) {
    if (num < pivot) less.push_back(num);
    else if (num == pivot) equal.push_back(num);
    else greater.push_back(num);
}

vector<int> sortedLess = quickSort(less);
vector<int> sortedGreater = quickSort(greater);

sortedLess.insert(sortedLess.end(), equal.begin(), equal.end());
sortedLess.insert(sortedLess.end(), sortedGreater.begin(), sortedGreater.end());
return sortedLess;
}

// Merge Sort
void merge(vector<int>& arr, int left, int mid, int right) {
    vector<int> temp;
    int i = left, j = mid + 1;
    while (i <= mid && j <= right) {
        if (arr[i] <= arr[j]) temp.push_back(arr[i++]);
        else temp.push_back(arr[j++]);
    }
    while (i <= mid) temp.push_back(arr[i++]);
    while (j <= right) temp.push_back(arr[j++]);

    for (int k = left; k <= right; k++) arr[k] = temp[k - left];
}

void mergeSort(vector<int>& arr, int left, int right) {
    if (left >= right) return;
    int mid = left + (right - left) / 2;
    mergeSort(arr, left, mid);
    mergeSort(arr, mid + 1, right);
    merge(arr, left, mid, right);
}

// Main Function
int main() {
    vector<int> arr = {170, 45, 75, 90, 802, 24, 2, 66};

    vector<int> arrRadix = arr;
    vector<int> arrQuick = arr;

```

```

vector<int> arrMerge = arr;

radixSort(arrRadix);
cout << "Radix Sorted Array: ";
for (int num : arrRadix) cout << num << " ";
cout << "\n";

vector<int> quickSorted = quickSort(arrQuick);
cout << "Quick Sorted Array: ";
for (int num : quickSorted) cout << num << " ";
cout << "\n";

mergeSort(arrMerge, 0, arrMerge.size() - 1);
cout << "Merge Sorted Array: ";
for (int num : arrMerge) cout << num << " ";
cout << "\n";

return 0;
}

```

Screenshot:

The screenshot shows a C++ IDE with a code editor and a terminal window. The code editor displays the following code:

```

72 void mergeSort(vector<int>& arr, int left, int right) {
73 }
74
75 // Main Function
76
77 Codeium: Refactor | Explain | X
78 int main() {
79     vector<int> arr = {170, 45, 75, 90, 802, 24, 2, 66};
80 }

```

The terminal window shows the following output:

```

PS C:\xampp\htdocs\rldho> cd "c:\xampp\htdocs\rldho\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) {
.\tempCodeRunnerFile }
Radix Sorted Array: 2 24 45 66 75 90 170 802
Quick Sorted Array: 2 24 45 66 75 90 170 802
Merge Sorted Array: 2 24 45 66 75 90 170 802
PS C:\xampp\htdocs\rldho> cd "c:\xampp\htdocs\rldho\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) {
.\tempCodeRunnerFile }

```