Nama : Frederico Steven Kwok

NPM : 242310037

Kelas : TI-24-PA1

Matkul: Lab. Analisis Algoritma dan Struktur Data

1. **Soal Queue**

```cpp
#include <iostream>
#include <cstring>
#include <limits>
using namespace std;

class Queue {
    private:
        int count;
        int MAX;
        string* data;

    public:
        Queue() {
            count = 0;
            MAX = 0;
        }

        void setCount(int x) {
            MAX = x;
            data = new string[MAX];
        }

        int size() {
            return count;
        }

        bool isEmpty() {
            if (count == 0) {
                return true;
            } else {
                return false;
```

```
            return false;
        }
    }

    bool isFull() {
        if (count >= MAX) {
            return true;
        } else {
            return false;
        }
    }

    void enqueue() {
        if (isFull()) {
            cout<<"Antrian Penuh...";
        } else {
            cin>>data[count];
            count++;
        }
    }

    int dequeue() {
        if (isEmpty()) {
            cout<<"Antrian Kosong."<<endl;
        } else {
            for (int a = 0; a < count-1; a++) {
                data[a] = data[a+1];
                cout<<data[a+1];
            }
            count--;
        }
    }

    int view() {
        if (isEmpty()) {
            cout<<"Antrian Kosong..."<<endl;
        } else {
            for (int a = count-1; a >= 0; a--) {
                cout<<data[a]<<endl;
            }
        }
        cout<<endl;
    }
};

int main() {
    int pilih;
    int n;
    string isi;
    string ans;
    Queue que;

    cout<<"Masukkan Jumlah Antrian : ";
    cin>>n;
    que.setCount(n);

    antri:
    cout<<"1. Masuk Antri "<<endl;

    cout<<"-> Pilih: ";
    cin>>pilih;
```

```
77      int pilih;
78      int n;
79      string isi;
80      string ans;
81      Queue que;
82
83      cout<<"Masukkan Jumlah Antrian : ";
84      cin>>n;
85      que.setCount(n);
86
87      antri:
88      cout<<"1. Masuk Antri "<<endl;
89
90      cout<<"-> Pilih: ";
91      cin>>pilih;
92
93      if (pilih == 1) {
94          for (int i = 0; i < n; i++) {
95              que.enqueue();
96          }
97      }
98
99      cout << "Ukuran Antrian: " << que.size() << endl;
100     cout << "Elemen Pertama: " << que.view() << endl;
101     cout << "Elemen Keluar: " << que.dequeue() << endl;
102     cout << "Elemen Pertama: " << que.view() << endl;
103     cout << "Ukuran Antrian: " << que.size() << endl;
104
105     return 0;
106 }
```

Output

```
Masukkan Jumlah Antrian : 3
1. Masuk Antri
-> Pilih: 1
12
32
14
Ukuran Antrian: 3
14
32
12

Elemen Pertama: 4745728
3214Elemen Keluar: 7339472
14
32

Elemen Pertama: 4745728
Ukuran Antrian: 2
```

2. **Soal Tree Manual Print**

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

class Node {
    public:
        int data;
        Node* left;
        Node* right;

        Node(int value) {
            data = value;
            left = NULL;
            right = NULL;
        }
};

class Tree {
    public:
        Node* root;
        Tree() {
            root = NULL;
        }

        void insert(int value) {
            root = insert(root, value);
        }

        Node* insert (Node* node, int value) {
            if (node == NULL) {
                node = new Node(value);
            } else if (value <= node->data) {
                node -> left = insert(node->left, value);
            } else {
                node -> right = insert(node->right, value);
            }

            return node;
        }

        void preorder() {
            preorder(root);
        }

        void preorder(Node* node) {
            if(node == NULL) return;
            cout << node->data << " ";
            preorder(node->left);
            preorder(node->right);
        }

        void postorder() {
```

```
51          void postorder() {
52              postorder(root);
53          }
54
55          void postorder(Node* node) {
56              if(node == NULL) return;
57              postorder(node->left);
58              postorder(node->right);
59              cout << node->data << " ";
60          }
61
62          void printTree(int mode) {
63              if (mode == 1) {
64                  cout<<"(NPM Gnajil) Pre-Order Traversal : ";
65                  preorder(root);
66                  cout<<endl;
67              }
68
69              else if (mode == 2) {
70                  cout<<"(NPM Genap) Post-Order Traversal : ";
71                  postorder(root);
72                  cout<<endl;
73              }
74          }
75  };
76
77  int main(){
78      Tree tree;
79      int n;
80
81      cout<<"Masukkan jumlah simpul : ";
82      cin>>n;
83
84
85      cout<<"Masukkan "<<n<<" nilai untuk Binary Tree: "<<endl;
86      for(int i = 0; i < n; i++) {
87          int data;
88          cout<<"Nilai ke-"<<i+1<<": ";
89          cin >> data;
90          tree.insert(data);
91      }
92
93      cout<<endl;
94      tree.printTree(1);
95      tree.printTree(2);
96
97      return 0;
98  }
```

Output

```
Masukkan jumlah simpul : 5
Masukkan 5 nilai untuk Binary Tree:
Nilai ke-1: 12
Nilai ke-2: 28
Nilai ke-3: 30
Nilai ke-4: 23
Nilai ke-5: 14

(NPM Gnajil) Pre-Order Traversal         : 12 28 23 14 30
(NPM Genap) Post-Order Traversal         : 14 23 30 28 12
```

3. **Soal graph multipath**

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

class Graph {
    public:


        void allPaths (int u, int d, bool visited[], int path[], int pathLength) {
            visited[u] = true;
            path[pathLength] = u;
            pathLength;

            if (u == d) {
                printPath(path, pathLength);
            } else {
                for (int i = 0; i < adjSize[u]; i++) {
                    int next = adj[u][i];
                    if (!visited[next]) {
                        allPaths(next, d, visited, path, pathLength);
                    }
                }
            }
            pathLength--;
            visited[u] = false;
        }

        void printGraph() {
            for (int u = 0; u < V; u++) {
                cout << u << " -> ";
            for (vector<int>::iterator it = adj[u].begin(); it != adj[u].end(); ++it) {
```

Output

Tidak ada output

4. **Soal double hashing**

```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <cmath>
using namespace std;

const int MAX = 100;
int storage[MAX];
int hdt_boundary;

class Hash {
  public:
    // Fungsi Mencari Bilangan Prima di Atas
    int prima_atas(int n) {
        if (n % 2 == 0)
            n = n + 1;
        else
            n = n + 2;

        bool ketemu = false;

        while (!ketemu) {
            bool prima = true;
            if (n % 2 == 0)
                prima = false;
            else {
                int i = 3;
                while (prima && i <= sqrt(n)) {
                    if (n % i == 0)
                        prima = false;
                    i += 2;
                    i += 2;
                }
            }

            if (prima)
                ketemu = true;
            else
                n = n + 2;
        }

        return n;
    }

    // Fungsi Mencari Bilangan Prima di Bawah (untuk hashing ke-2)
    int prima_bawah() {
        int n = hdt_boundary - 1;
        while (n > 1) {
            bool prima = true;
            for (int i = 2; i <= sqrt(n); ++i) {
                if (n % i == 0) {
                    prima = false;
                    break;
                }
            }
            if (prima) return n;
            n--;
        }
        return 2;
    }
    // Fungsi Mencari Bilangan Prima di Atas
        void tambah_double_hashing(int n) {
```

```
        void tambah_double_hashing(int n) {
            int hash;
            int hash2 = prima_bawah();
            bool inserted = false;
            int i = 0;

        while (!inserted && i < hdt_boundary) {
            hash = ((n % hdt_boundary) + i * ((n % hash2) + 1)) % hdt_boundary;

            if (storage[hash] == 0) {
                storage[hash] = n;
                inserted = true;
            } else {
                ++i;
                cout << "Terjadi tabrakan di " << hash << endl;
            }
        }

        if (i == 0)
            cout << "Langsung" << endl;

        cout << "Isi hash[" << hash << "] dengan " << n << endl;
        cout << "=========================================" << endl;
    }

// Fungsi Cetak Output Hashtable
    void cetak(int n) {
        cout << endl;
        cout << "Output program : " << endl;
        for (int a = 0; a < n; ++a) {
            cout << "hash[" << a << "] = " << storage[a] << endl;
        }
    }
};

// Main Program
int main() {
    Hash h;
    int n, nilai;
    char pilihan;
    string cara;

    cout << "Masukan jumlah data : ";
    cin >> n;

    cout << endl << endl << "Proses pemasukan data ke hashtable " << endl;

    hdt_boundary = h.prima_atas(n);

    for (int a = 0; a < n; ++a) {
        cin>>nilai;
            h.tambah_double_hashing(nilai);
        }
    }

    // Output hasil akhir
    h.cetak(n);

}
```

Output

```
Masukan jumlah data : 5


Proses pemasukan data ke hashtable
12
0
32
7
23

Output program :
hash[0] = 7
hash[1] = 0
hash[2] = 23
hash[3] = 0
hash[4] = 32
```