

SURAT PERNYATAAN PESERTA UJIAN

Dengan ini saya mahasiswa peserta Ujian Akhir Semester (UAS) Semester Genap Tahun Akademik 2023/2024 Universitas Dian Nusantara, Jakarta :

Nama : Muhamad Aditya Saputra
NIM : 411231139
Program Studi : Teknik Informatika

Menyatakan sebagai berikut :

1. Saya akan melaksanakan Ujian Akhir Semester secara *online* sesuai jadwal yang ditetapkan di SISKAS.
2. Saya berjanji mentaati tata tertib ujian dengan bersikap jujur, tidak menyontek, tidak meng-*copy paste* jawaban ujian orang lain dan tidak melakukan praktek perjokian.
3. Saya siap menerima sanksi nilai E (tidak lulus) apabila melakukan pelanggaran ujian sesuai dengan butir 2.
4. Bahwa saya telah memahami kewajiban pembayaran uang kuliah sesuai dengan ketentuan yang berlaku.

Demikian pernyataan ini saya buat dengan sesungguhnya untuk dapat dipergunakan sebagaimana mestinya.

Bekasi, 27 Juli 2024

(Muhamad Aditya Saputra)

Hasil Kerja Ujian Akhir Semester Semester Genap 2023/2024

Nama : Muhamad Aditya Saputra

NIM : 411231139

No. HP : 085155225938

Waktu Ujian :

Mata Kuliah : Struktur Data dan Algoritma

Nama Dosen : Rani Laple Satria Putra, S.Kom, M.Kom



Fakultas :

Program Studi :

TATAP MUKA

16

Kode Mata Kuliah :

Kode Kelas :

Nomor 1

```
#include <iostream>
#include <string>

using namespace std;

// Struktur data untuk menyimpan informasi mahasiswa
struct Mahasiswa {
    string nama;
    string npm;
    string jurusan;
};

// Kelas Queue untuk menyimpan data mahasiswa
class Queue {
private:
    Mahasiswa* data;
    int front;
    int rear;
    int kapasitas;
    int jumlah;

public:
    // Konstruktor untuk inisialisasi queue
    Queue(int kapasitas) {
        this->kapasitas = kapasitas;
        this->data = new Mahasiswa[kapasitas];
        this->front = 0;
        this->rear = 0;
        this->jumlah = 0;
    }

    // Destruktor untuk menghapus memori yang dialokasikan
    ~Queue() {
        delete[] data;
    }

    // Fungsi untuk menambah elemen ke queue (enqueue)
    void enqueue(Mahasiswa mahasiswa) {
        if (jumlah < kapasitas) {
            data[rear] = mahasiswa;
            rear = (rear + 1) % kapasitas;
            jumlah++;
            cout << "Mahasiswa " << mahasiswa.nama << " berhasil ditambahkan ke queue." << endl;
        } else {
            cout << "Queue sudah penuh. Tidak dapat menambahkan mahasiswa lagi." << endl;
        }
    }
};
```



```

    }
}

// Fungsi untuk menghapus elemen dari queue (dequeue)
void dequeue() {
    if (jumlah > 0) {
        Mahasiswa mahasiswa = data[front];
        front = (front + 1) % kapasitas;
        jumlah--;
        cout << "Mahasiswa " << mahasiswa.nama << " berhasil dihapus dari queue." << endl;
    } else {
        cout << "Queue kosong. Tidak dapat menghapus mahasiswa." << endl;
    }
}

// Fungsi untuk menampilkan elemen di depan queue (front)
void tampilkanFront() {
    if (jumlah > 0) {
        Mahasiswa mahasiswa = data[front];
        cout << "Mahasiswa di depan queue: " << mahasiswa.nama << " (" << mahasiswa.npm << ")" << endl;
    } else {
        cout << "Queue kosong." << endl;
    }
}

// Fungsi untuk menampilkan elemen di belakang queue (rear)
void tampilkanRear() {
    if (jumlah > 0) {
        Mahasiswa mahasiswa = data[(rear - 1 + kapasitas) % kapasitas];
        cout << "Mahasiswa di belakang queue: " << mahasiswa.nama << " (" << mahasiswa.npm << ")" << endl;
    } else {
        cout << "Queue kosong." << endl;
    }
}

};

int main() {
    // Buat queue dengan kapasitas 5
    Queue queue(5);

    // Tambah mahasiswa ke queue
    Mahasiswa mahasiswa1 = {"John Doe", "1234567890", "Informatika"};
    queue.enqueue(mahasiswa1);

    Mahasiswa mahasiswa2 = {"Muhamad Aditya Saputra", "9876543210", "Informatika"};
    queue.enqueue(mahasiswa2);

    // Tampilkan elemen di depan queue
    queue.tampilkanFront();
}

```



```
// Tampilkan elemen di belakang queue
queue.tampilkanRear();
```

```
// Hapus mahasiswa dari queue
queue.dequeue();
```

```
// Tampilkan elemen di depan queue
queue.tampilkanFront();
```

```
return 0;
```

```
}
```

```
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 // Struktur data untuk menyimpan informasi mahasiswa
7 struct Mahasiswa {
8     string nama;
9     string rpg;
10    string jurusan;
11 };
12
13 // Kelas Queue untuk menyimpan data mahasiswa
14 class Queue {
15 private:
16     Mahasiswa* data;
17     int front;
18     int rear;
19     int kapasitas;
20     int jumlah;
21
22 public:
23     // Konstruktor untuk inisialisasi queue
24     Queue(int kapasitas) {
25         this->kapasitas = kapasitas;
26         this->data = new Mahasiswa[kapasitas];
27         this->front = 0;
28         this->rear = 0;
29         this->jumlah = 0;
30     }
31
32     // Destruktor untuk menghapus memori yang dialokasikan
33     ~Queue() {
34         delete[] data;
35     }
36
37     // Fungsi untuk menambah elemen ke queue (enqueue)
38     void enqueue(Mahasiswa mahasiswa) {
39         if (jumlah < kapasitas) {
40             data[rear] = mahasiswa;
41             rear = (rear + 1) % kapasitas;
42             jumlah++;
43             cout << "Mahasiswa " << mahasiswa.nama << " berhasil ditambahkan ke queue." << endl;
44         } else {
45             cout << "Queue sudah penuh. Tidak dapat menambahkan mahasiswa lagi." << endl;
46         }
47     }
48
49     // Fungsi untuk menghapus elemen dari queue (dequeue)
50     void dequeue() {
51         if (jumlah > 0) {
52             Mahasiswa mahasiswa = data[front];
53             front = (front + 1) % kapasitas;
54             jumlah--;
55             cout << "Mahasiswa " << mahasiswa.nama << " berhasil dihapus dari queue." << endl;
56         } else {
57             cout << "Queue kosong. Tidak dapat menghapus mahasiswa." << endl;
58         }
59     }
60
61     // Fungsi untuk menampilkan elemen di depan queue (front)
62     void tampilkanFront() {
63         if (jumlah > 0) {
64             Mahasiswa mahasiswa = data[front];
65             cout << "Mahasiswa di depan queue: " << mahasiswa.nama << " (" << mahasiswa.rpg << ") " << endl;
66         } else {
67             cout << "Queue kosong." << endl;
68         }
69     }
70
71     // Fungsi untuk menampilkan elemen di belakang queue (rear)
72     void tampilkanRear() {
73         if (jumlah > 0) {
74             Mahasiswa mahasiswa = data[(rear - 1 + kapasitas) % kapasitas];
75             cout << "Mahasiswa di belakang queue: " << mahasiswa.nama << " (" << mahasiswa.rpg << ") " << endl;
76         } else {
77             cout << "Queue kosong." << endl;
78         }
79     }
80 };
81
82 int main() {
83     // Buat queue dengan kapasitas 5
84     Queue queue(5);
85
86     // Tambah mahasiswa ke queue
87     Mahasiswa mahasiswa1 = {"John Doe", "123456789", "Informatika"};
88     queue.enqueue(mahasiswa1);
89
90     Mahasiswa mahasiswa2 = {"Muhammad Aditya Saputra", "987654321", "Informatika"};
91     queue.enqueue(mahasiswa2);
92
93     // Tampilkan elemen di depan queue
94     queue.tampilkanFront();
95
96     // Tampilkan elemen di belakang queue
97     queue.tampilkanRear();
98
99     // Hapus mahasiswa dari queue
100    queue.dequeue();
101
102    // Tampilkan elemen di depan queue
103    queue.tampilkanFront();
104
105    return 0;
106 }
```



Nomor 2

```
#include <iostream>
#include <string>

using namespace std;

// Struktur data untuk menyimpan informasi kontak
struct Kontak {
    string nama;
    string nomorTelepon;
};

// Fungsi untuk mencari nomor telepon berdasarkan nama
void cariNomorTelepon(Kontak kontak[], int jumlahKontak, string nama) {
    bool ditemukan = false;
    for (int i = 0; i < jumlahKontak; i++) {
        if (kontak[i].nama == nama) {
            cout << "Nomor telepon " << nama << " adalah " << kontak[i].nomorTelepon << endl;
            ditemukan = true;
            break;
        }
    }
    if (!ditemukan) {
        cout << "Nomor telepon " << nama << " tidak ditemukan!" << endl;
    }
}

int main() {
    // Buat array kontak dengan 5 elemen
    Kontak kontak[5];

    // Isi array kontak
    kontak[0] = {"John Doe", "081234567890"};
    kontak[1] = {"Muhamad Aditya Saputra", "085678912345"};
    kontak[2] = {"Jane Doe", "081987654321"};
    kontak[3] = {"Budi Setiawan", "085432109876"};
    kontak[4] = {"Rudi Hartono", "081765432109"};

    // Cari nomor telepon berdasarkan nama
    string nama;
    cout << "Masukkan nama: ";
    getline(cin, nama);
    cariNomorTelepon(kontak, 5, nama);

    return 0;
}
```





```
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  // Struktur data untuk menyimpan informasi kontak
7  struct Kontak {
8      string nama;
9      string nomorTelepon;
10 };
11
12 // Fungsi untuk mencari nomor telepon berdasarkan nama
13 void cariNomorTelepon(Kontak kontak[], int jumlahKontak, string nama) {
14     bool ditemukan = false;
15     for (int i = 0; i < jumlahKontak; i++) {
16         if (kontak[i].nama == nama) {
17             cout << "Nomor telepon " << nama << " adalah " << kontak[i].nomorTelepon << endl;
18             ditemukan = true;
19             break;
20         }
21     }
22     if (!ditemukan) {
23         cout << "Nomor telepon " << nama << " tidak ditemukan!" << endl;
24     }
25 }
26
27 int main() {
28     // Buat array kontak dengan 5 elemen
29     Kontak kontak[5];
30
31     // Isi array kontak
32     kontak[0] = {"John Doe", "081234567890"};
33     kontak[1] = {"Muhamad Aditya Saputra", "085678912345"};
34     kontak[2] = {"Jane Doe", "081987654321"};
35     kontak[3] = {"Budi Setiawan", "085432109876"};
36     kontak[4] = {"Rudi Hartono", "081765432109"};
37
38     // Cari nomor telepon berdasarkan nama
39     string nama;
40     cout << "Masukkan nama: ";
41     getline(cin, nama);
42     cariNomorTelepon(kontak, 5, nama);
43
44     return 0;
45 }
```



Nomor 3

```
#include <iostream>
#include <string>

using namespace std;

struct Transaksi {
    string waktu;
    string jenisPembayaran;
    int jumlah;
};

void selectionSort(Transaksi arr[], int n, bool sortByWaktu) {
    for (int i = 0; i < n - 1; i++) {
        int minIndex = i;
        for (int j = i + 1; j < n; j++) {
            if (sortByWaktu) {
                if (arr[j].waktu < arr[minIndex].waktu) {
                    minIndex = j;
                }
            } else {
                if (arr[j].jenisPembayaran < arr[minIndex].jenisPembayaran) {
                    minIndex = j;
                }
            }
        }
        swap(arr[i], arr[minIndex]);
    }
}

void printTransaksi(Transaksi arr[], int n) {
    cout << "No.\tWaktu\tJenis Pembayaran\tJumlah" << endl;
    for (int i = 0; i < n; i++) {
        cout << i + 1 << "\t" << arr[i].waktu << "\t" << arr[i].jenisPembayaran << "\t" << arr[i].jumlah << endl;
    }
}

int main() {
    int n;
    cout << "Masukkan jumlah transaksi: ";
    cin >> n;

    Transaksi transaksi[n];

    for (int i = 0; i < n; i++) {
        cout << "Masukkan waktu transaksi " << i + 1 << ": ";
        cin >> transaksi[i].waktu;
        cout << "Masukkan jenis pembayaran transaksi " << i + 1 << ": ";
    }
}
```




```

        cin >> transaksi[i].jenisPembayaran;
        cout << "Masukkan jumlah transaksi " << i + 1 << ": ";
        cin >> transaksi[i].jumlah;
    }

    cout << "Pilih metode pengurutan:" << endl;
    cout << "1. Waktu" << endl;
    cout << "2. Jenis Pembayaran" << endl;
    int pilihan;
    cin >> pilihan;

    if (pilihan == 1) {
        selectionSort(transaksi, n, true);
    } else if (pilihan == 2) {
        selectionSort(transaksi, n, false);
    } else {
        cout << "Pilihan tidak valid" << endl;
        return 1;
    }

    printTransaksi(transaksi, n);

    return 0;
}

```



```

1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  struct Transaksi {
7      string waktu;
8      string jenisPembayaran;
9      int jumlah;
10 };
11
12 void selectionSort(Transaksi arr[], int n, bool sortByWaktu) {
13     for (int i = 0; i < n - 1; i++) {
14         int minIndex = i;
15         for (int j = i + 1; j < n; j++) {
16             if (sortByWaktu) {
17                 if (arr[j].waktu < arr[minIndex].waktu) {
18                     minIndex = j;
19                 }
20             } else {
21                 if (arr[j].jenisPembayaran < arr[minIndex].jenisPembayaran) {
22                     minIndex = j;
23                 }
24             }
25         }
26         swap(arr[i], arr[minIndex]);
27     }
28 }
29
30 void printTransaksi(Transaksi arr[], int n) {
31     cout << "No.\tWaktu\tJenis Pembayaran\tJumlah" << endl;
32     for (int i = 0; i < n; i++) {
33         cout << i + 1 << "\t" << arr[i].waktu << "\t" << arr[i].jenisPembayaran << "\t" << arr[i].jumlah << endl;
34     }
35 }
36
37 int main() {
38     int n;
39     cout << "Masukkan jumlah transaksi: ";
40     cin >> n;
41
42     Transaksi transaksi[n];
43
44     for (int i = 0; i < n; i++) {
45         cout << "Masukkan waktu transaksi " << i + 1 << ": ";
46         cin >> transaksi[i].waktu;
47         cout << "Masukkan jenis pembayaran transaksi " << i + 1 << ": ";
48         cin >> transaksi[i].jenisPembayaran;
49         cout << "Masukkan jumlah transaksi " << i + 1 << ": ";
50         cin >> transaksi[i].jumlah;
51     }
52
53     cout << "Pilih metode pengurutan:" << endl;
54     cout << "1. Waktu" << endl;
55     cout << "2. Jenis Pembayaran" << endl;
56     int pilihan;
57     cin >> pilihan;
58
59     if (pilihan == 1) {
60         selectionSort(transaksi, n, true);
61     } else if (pilihan == 2) {
62         selectionSort(transaksi, n, false);
63     } else {
64         cout << "Pilihan tidak valid" << endl;
65         return 1;
66     }
67
68     printTransaksi(transaksi, n);
69
70     return 0;
71 }

```



Nomor 4

```
#include <iostream>
#include <string>

using namespace std;

// Struktur data untuk node binary tree
struct Node {
    int noRekening;
    string namaNasabah;
    Node* left;
    Node* right;
};

// Fungsi untuk membuat node baru
Node* createNode(int noRekening, string namaNasabah) {
    Node* newNode = new Node();
    newNode->noRekening = noRekening;
    newNode->namaNasabah = namaNasabah;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

// Fungsi untuk memasukkan data ke dalam binary tree
Node* insertNode(Node* root, int noRekening, string namaNasabah) {
    if (root == NULL) {
        root = createNode(noRekening, namaNasabah);
    } else if (noRekening < root->noRekening) {
        root->left = insertNode(root->left, noRekening, namaNasabah);
    } else if (noRekening > root->noRekening) {
        root->right = insertNode(root->right, noRekening, namaNasabah);
    }
    return root;
}

// Fungsi untuk mencari data dalam binary tree
Node* searchNode(Node* root, int noRekening) {
    if (root == NULL || root->noRekening == noRekening) {
        return root;
    } else if (noRekening < root->noRekening) {
        return searchNode(root->left, noRekening);
    } else {
        return searchNode(root->right, noRekening);
    }
}

// Fungsi untuk menghapus data dari binary tree
Node* deleteNode(Node* root, int noRekening) {
```



```

if (root == NULL) {
    return root;
} else if (noRekening < root->noRekening) {
    root->left = deleteNode(root->left, noRekening);
} else if (noRekening > root->noRekening) {
    root->right = deleteNode(root->right, noRekening);
} else {
    if (root->left == NULL) {
        Node* temp = root->right;
        delete root;
        return temp;
    } else if (root->right == NULL) {
        Node* temp = root->left;
        delete root;
        return temp;
    } else {
        Node* temp = root->right;
        while (temp->left != NULL) {
            temp = temp->left;
        }
        root->noRekening = temp->noRekening;
        root->namaNasabah = temp->namaNasabah;
        root->right = deleteNode(root->right, temp->noRekening);
    }
}
return root;
}

// Fungsi untuk menampilkan data dalam binary tree
void displayTree(Node* root) {
    if (root != NULL) {
        displayTree(root->left);
        cout << "No. Rekening: " << root->noRekening << ", Nama Nasabah: " << root->namaNasabah << endl;
        displayTree(root->right);
    }
}

int main() {
    Node* root = NULL;

    // Memasukkan data ke dalam binary tree
    root = insertNode(root, 12345, "John Doe");
    root = insertNode(root, 67890, "Jane Doe");
    root = insertNode(root, 34567, "Bob Smith");

    // Menampilkan data dalam binary tree
    cout << "Data dalam binary tree:" << endl;
    displayTree(root);

    // Mencari data dalam binary tree

```



```

int noRekening = 12345;
Node* foundNode = searchNode(root, noRekening);
if (foundNode != NULL) {
    cout << "Data nasabah dengan no. rekening " << noRekening << " ditemukan:" << endl;
    cout << "Nama Nasabah: " << foundNode->namaNasabah << endl;
} else {
    cout << "Data nasabah dengan no. rekening " << noRekening << " tidak ditemukan." << endl;
}

// Menghapus data dari binary tree
noRekening = 67890;
root = deleteNode(root, noRekening);

// Menampilkan data dalam binary tree setelah penghapusan
cout << "Data dalam binary tree setelah penghapusan:" << endl;
displayTree(root);

return 0;
}

```



```

1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 // Struktur data untuk node binary tree
7 struct Node {
8     int noRekening;
9     string namaNasabah;
10    Node* left;
11    Node* right;
12 };
13
14 // Fungsi untuk membuat node baru
15 Node* createNode(int noRekening, string namaNasabah) {
16     Node* newNode = new Node();
17     newNode->noRekening = noRekening;
18     newNode->namaNasabah = namaNasabah;
19     newNode->left = NULL;
20     newNode->right = NULL;
21     return newNode;
22 }
23
24 // Fungsi untuk memasukkan data ke dalam binary tree
25 Node* insertNode(Node* root, int noRekening, string namaNasabah) {
26     if (root == NULL) {
27         root = createNode(noRekening, namaNasabah);
28     } else if (noRekening < root->noRekening) {
29         root->left = insertNode(root->left, noRekening, namaNasabah);
30     } else if (noRekening > root->noRekening) {
31         root->right = insertNode(root->right, noRekening, namaNasabah);
32     }
33     return root;
34 }
35
36 // Fungsi untuk mencari data dalam binary tree
37 Node* searchNode(Node* root, int noRekening) {
38     if (root == NULL || root->noRekening == noRekening) {
39         return root;
40     } else if (noRekening < root->noRekening) {
41         return searchNode(root->left, noRekening);
42     } else {
43         return searchNode(root->right, noRekening);
44     }
45 }
46
47 // Fungsi untuk menghapus data dari binary tree
48 Node* deleteNode(Node* root, int noRekening) {
49     if (root == NULL) {
50         return root;
51     } else if (noRekening < root->noRekening) {
52         root->left = deleteNode(root->left, noRekening);
53     } else if (noRekening > root->noRekening) {
54         root->right = deleteNode(root->right, noRekening);
55     } else {
56         if (root->left == NULL) {
57             Node* temp = root->right;
58             delete root;
59             return temp;
60         } else if (root->right == NULL) {
61             Node* temp = root->left;
62             delete root;
63             return temp;
64         } else {
65             Node* temp = root->right;
66             while (temp->left != NULL) {
67                 temp = temp->left;
68             }
69             root->noRekening = temp->noRekening;
70             root->namaNasabah = temp->namaNasabah;
71             root->right = deleteNode(root->right, temp->noRekening);
72         }
73     }
74     return root;
75 }
76
77 // Fungsi untuk menampilkan data dalam binary tree
78 void displayTree(Node* root) {
79     if (root != NULL) {
80         displayTree(root->left);
81         cout << "No. Rekening: " << root->noRekening << ", Nama Nasabah: " << root->namaNasabah << endl;
82         displayTree(root->right);
83     }
84 }
85
86 int main() {
87     Node* root = NULL;
88
89     // Memasukkan data ke dalam binary tree
90     root = insertNode(root, 12345, "John Doe");
91     root = insertNode(root, 67890, "Jane Doe");
92     root = insertNode(root, 34567, "Bob Smith");
93
94     // Menampilkan data dalam binary tree
95     cout << "Data dalam binary tree:" << endl;
96     displayTree(root);
97
98     // Mencari data dalam binary tree
99     int noRekening = 12345;
100    Node* foundNode = searchNode(root, noRekening);
101    if (foundNode != NULL) {
102        cout << "Data nasabah dengan no. rekening " << noRekening << " ditemukan:" << endl;
103        cout << "Nama Nasabah: " << foundNode->namaNasabah << endl;
104    } else {
105        cout << "Data nasabah dengan no. rekening " << noRekening << " tidak ditemukan." << endl;
106    }
107
108    // Menghapus data dari binary tree
109    noRekening = 67890;
110    root = deleteNode(root, noRekening);
111
112    // Menampilkan data dalam binary tree setelah penghapusan
113    cout << "Data dalam binary tree setelah penghapusan:" << endl;
114    displayTree(root);
115
116    return 0;
117 }

```



Nomor 5

```
#include <iostream>
#include <string>

using namespace std;

struct Transaksi {
    string waktu;
    string jenisPembayaran;
    int jumlah;
};

void selectionSort(Transaksi arr[], int n, bool sortByWaktu) {
    for (int i = 0; i < n - 1; i++) {
        int minIndex = i;
        for (int j = i + 1; j < n; j++) {
            if (sortByWaktu) {
                if (arr[j].waktu < arr[minIndex].waktu) {
                    minIndex = j;
                }
            } else {
                if (arr[j].jenisPembayaran < arr[minIndex].jenisPembayaran) {
                    minIndex = j;
                }
            }
        }
        swap(arr[i], arr[minIndex]);
    }
}

void printTransaksi(Transaksi arr[], int n) {
    cout << "No.\tWaktu\tJenis Pembayaran\tJumlah" << endl;
    for (int i = 0; i < n; i++) {
        cout << i + 1 << "\t" << arr[i].waktu << "\t" << arr[i].jenisPembayaran << "\t" << arr[i].jumlah << endl;
    }
}

int main() {
    int n;
    cout << "Masukkan jumlah transaksi: ";
    cin >> n;

    Transaksi transaksi[n];

    for (int i = 0; i < n; i++) {
        cout << "Masukkan waktu transaksi " << i + 1 << ": ";
        cin >> transaksi[i].waktu;
        cout << "Masukkan jenis pembayaran transaksi " << i + 1 << ": ";
        cin >> transaksi[i].jenisPembayaran;
    }
}
```



```

        cout << "Masukkan jumlah transaksi " << i + 1 << ": ";
        cin >> transaksi[i].jumlah;
    }

    cout << "Pilih metode pengurutan:" << endl;
    cout << "1. Waktu" << endl;
    cout << "2. Jenis Pembayaran" << endl;
    int pilihan;
    cin >> pilihan;

    if (pilihan == 1) {
        selectionSort(transaksi, n, true);
    } else if (pilihan == 2) {
        selectionSort(transaksi, n, false);
    } else {
        cout << "Pilihan tidak valid" << endl;
        return 1;
    }

    printTransaksi(transaksi, n);

    return 0;
}

```




```

1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  struct Transaksi {
7      string waktu;
8      string jenisPembayaran;
9      int jumlah;
10 };
11
12 void selectionSort(Transaksi arr[], int n, bool sortByWaktu) {
13     for (int i = 0; i < n - 1; i++) {
14         int minIndex = i;
15         for (int j = i + 1; j < n; j++) {
16             if (sortByWaktu) {
17                 if (arr[j].waktu < arr[minIndex].waktu) {
18                     minIndex = j;
19                 }
20             } else {
21                 if (arr[j].jenisPembayaran < arr[minIndex].jenisPembayaran) {
22                     minIndex = j;
23                 }
24             }
25         }
26         swap(arr[i], arr[minIndex]);
27     }
28 }
29
30 void printTransaksi(Transaksi arr[], int n) {
31     cout << "No.\tWaktu\tJenis Pembayaran\tJumlah" << endl;
32     for (int i = 0; i < n; i++) {
33         cout << i + 1 << "\t" << arr[i].waktu << "\t" << arr[i].jenisPembayaran << "\t" << arr[i].jumlah << endl;
34     }
35 }
36
37 int main() {
38     int n;
39     cout << "Masukkan jumlah transaksi: ";
40     cin >> n;
41
42     Transaksi transaksi[n];
43
44     for (int i = 0; i < n; i++) {
45         cout << "Masukkan waktu transaksi " << i + 1 << ": ";
46         cin >> transaksi[i].waktu;
47         cout << "Masukkan jenis pembayaran transaksi " << i + 1 << ": ";
48         cin >> transaksi[i].jenisPembayaran;
49         cout << "Masukkan jumlah transaksi " << i + 1 << ": ";
50         cin >> transaksi[i].jumlah;
51     }
52
53     cout << "Pilih metode pengurutan:" << endl;
54     cout << "1. Waktu" << endl;
55     cout << "2. Jenis Pembayaran" << endl;
56     int pilihan;
57     cin >> pilihan;
58
59     if (pilihan == 1) {
60         selectionSort(transaksi, n, true);
61     } else if (pilihan == 2) {
62         selectionSort(transaksi, n, false);
63     } else {
64         cout << "Pilihan tidak valid" << endl;
65         return 1;
66     }
67
68     printTransaksi(transaksi, n);
69
70     return 0;
71 }

```

