**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 SISKA.
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

**Kode Mata Kuliah :**

**Kode Kelas :**

**Fakultas :**

**Program Studi :**

**TATAP MUKA**

**16**

**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;

}

A screen shot of a computer program

Description automatically generated

**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;

}

**A screen shot of a computer program

Description automatically generated**

**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;

}

A screen shot of a computer program

Description automatically generated

**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;

}

A screen shot of a computer screen

Description automatically generated

**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;

}

A screen shot of a computer program

Description automatically generated