LAPORAN PRAKTIKUM

POSTTEST 6

ALGORITMA PEMROGRAMAN LANJUT

Disusun oleh:

Reza Alameka (2409106107)

Kelas (C2 ‘24)

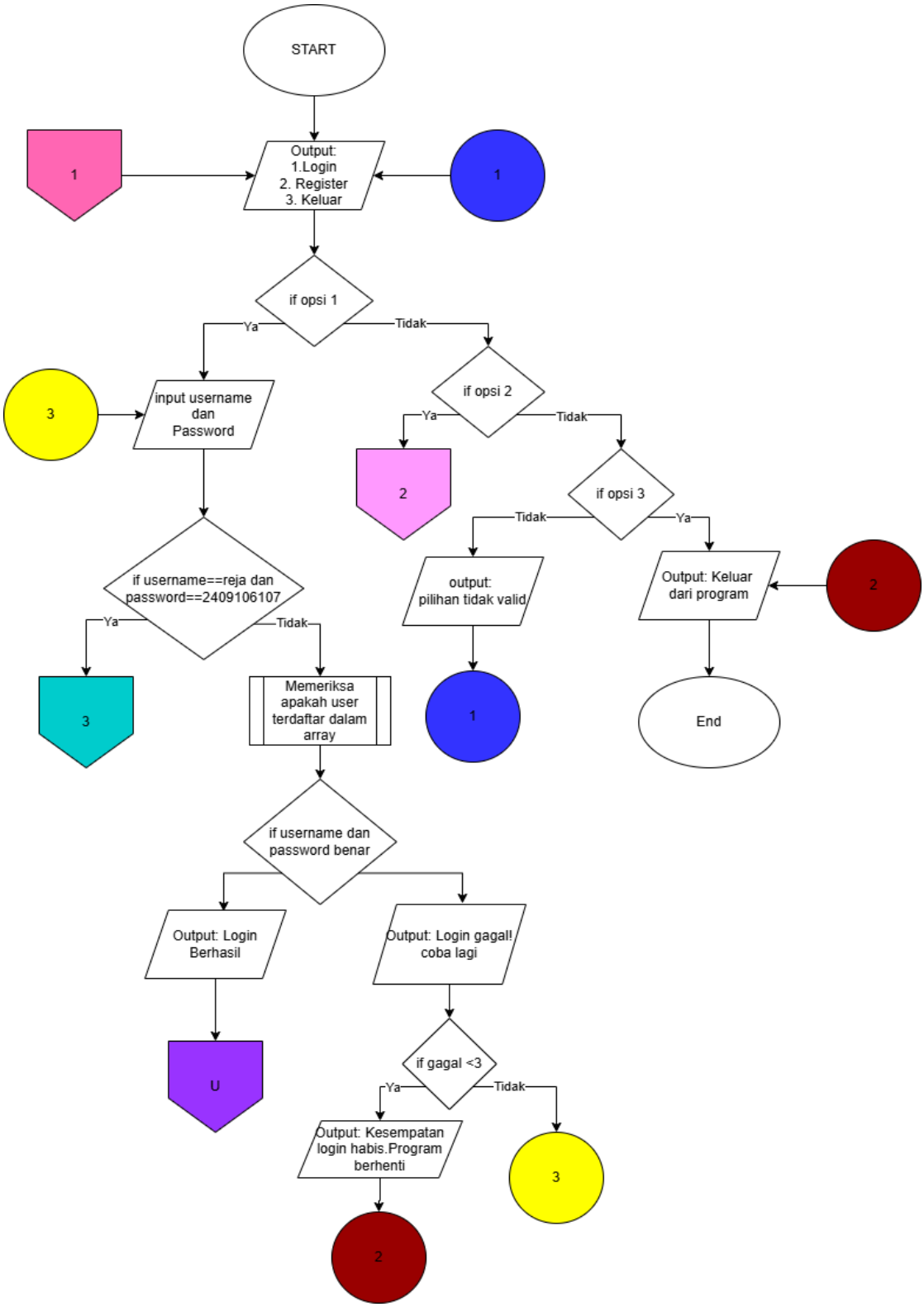
PROGRAM STUDI INFORMATIKA

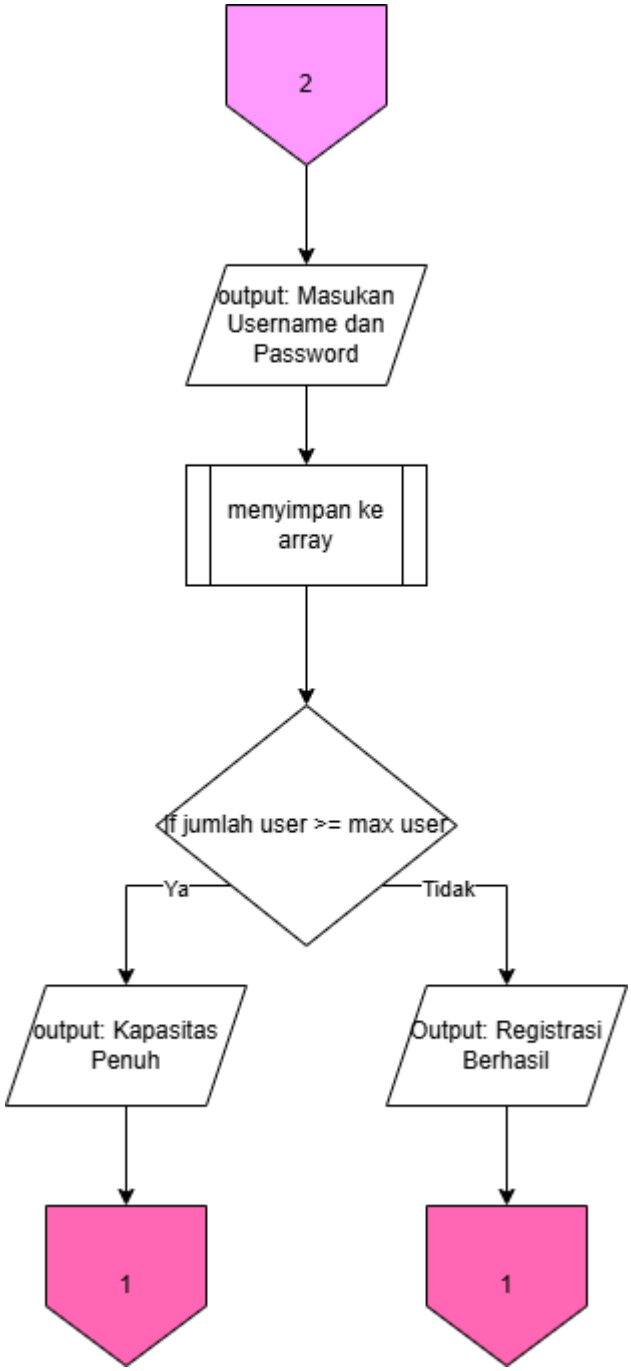
UNIVERSITAS MULAWARMAN

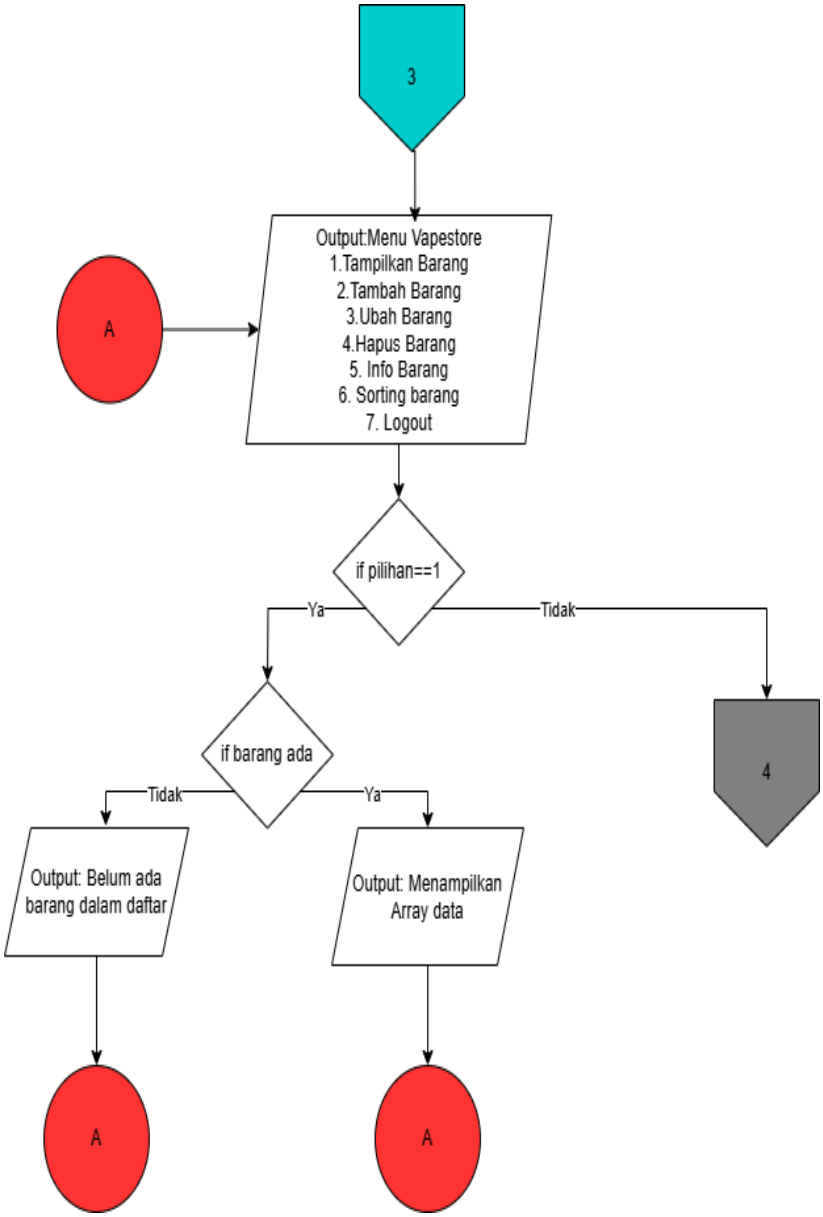
SAMARINDA

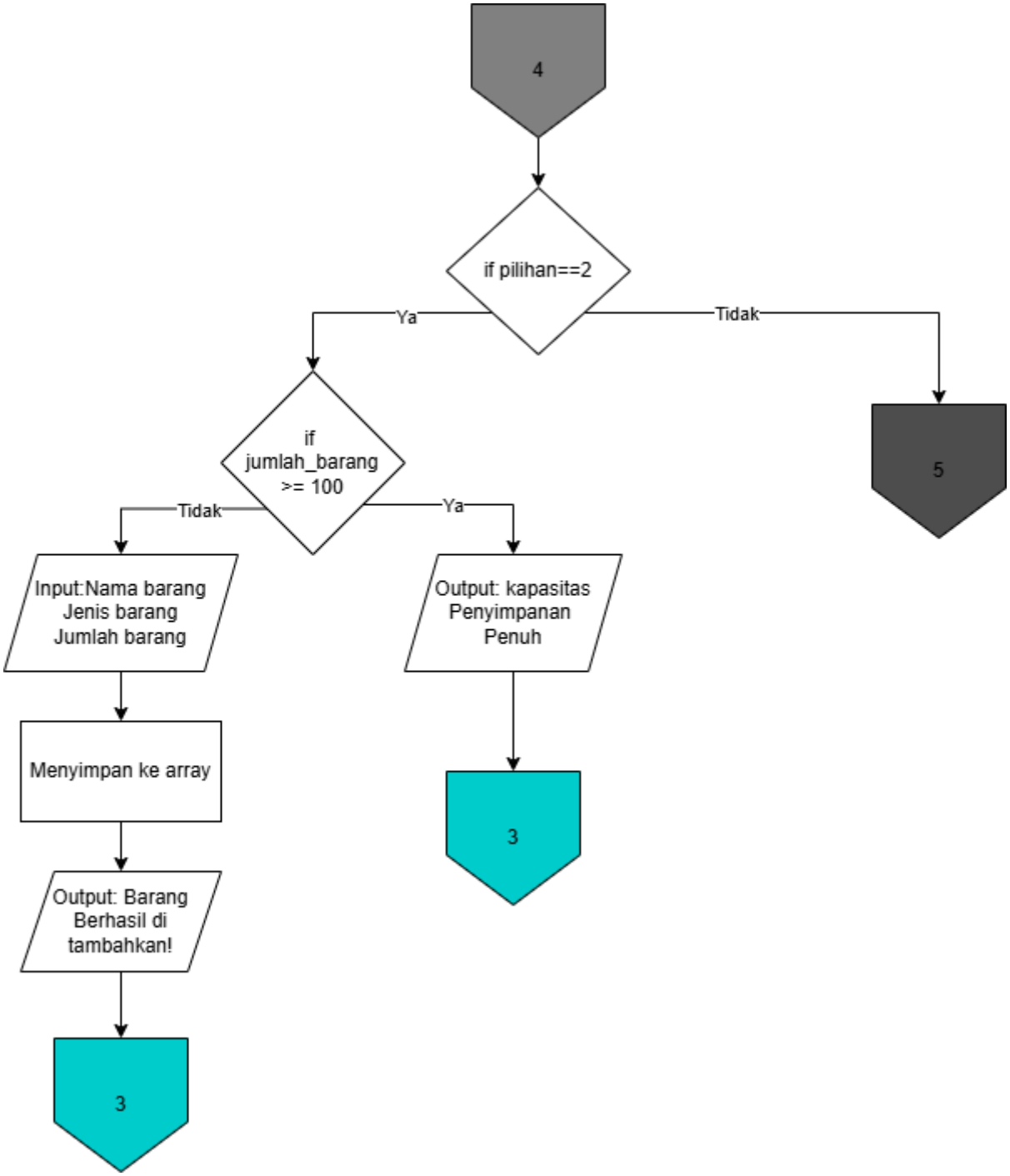
2025

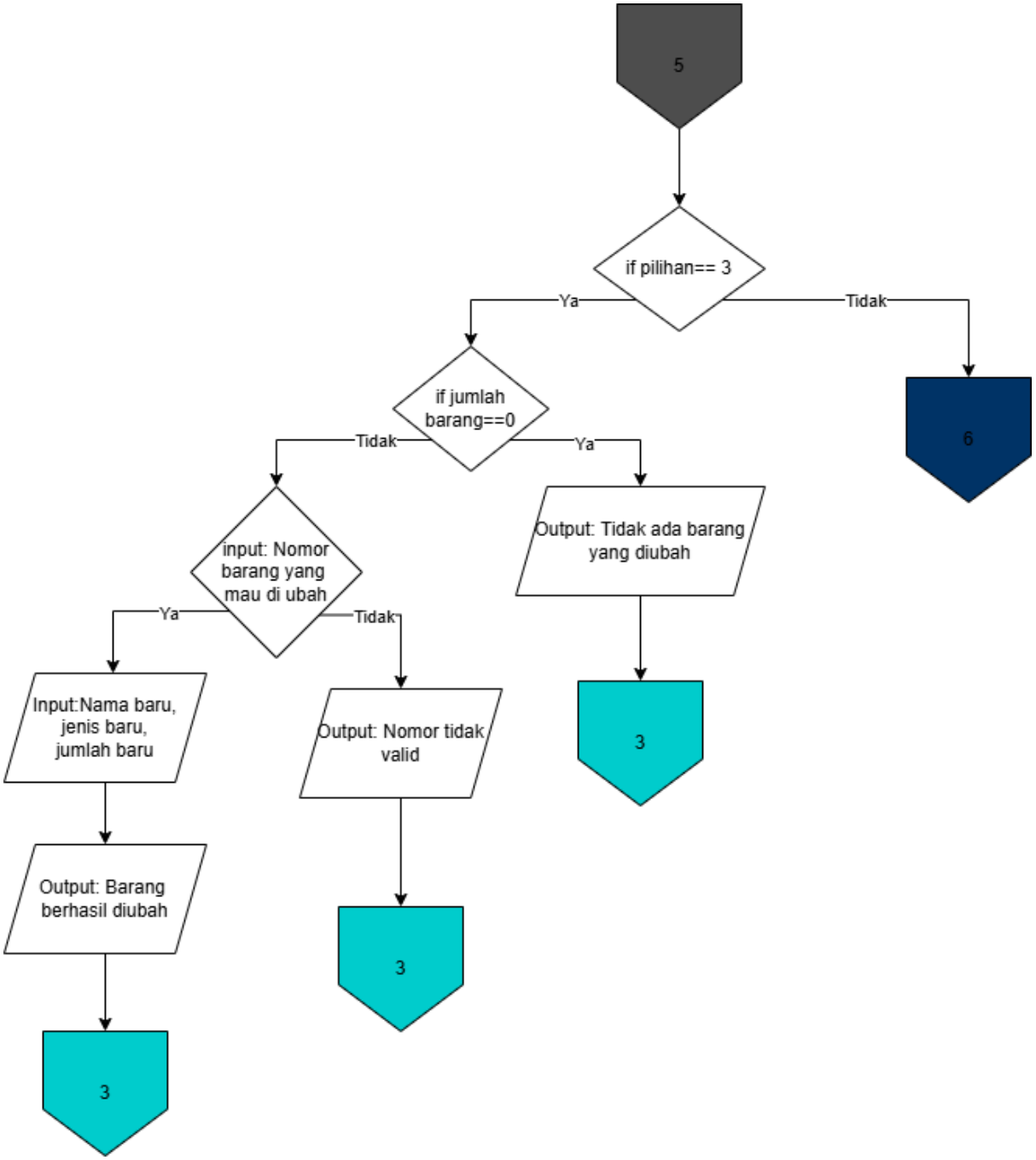
1. Flowchart

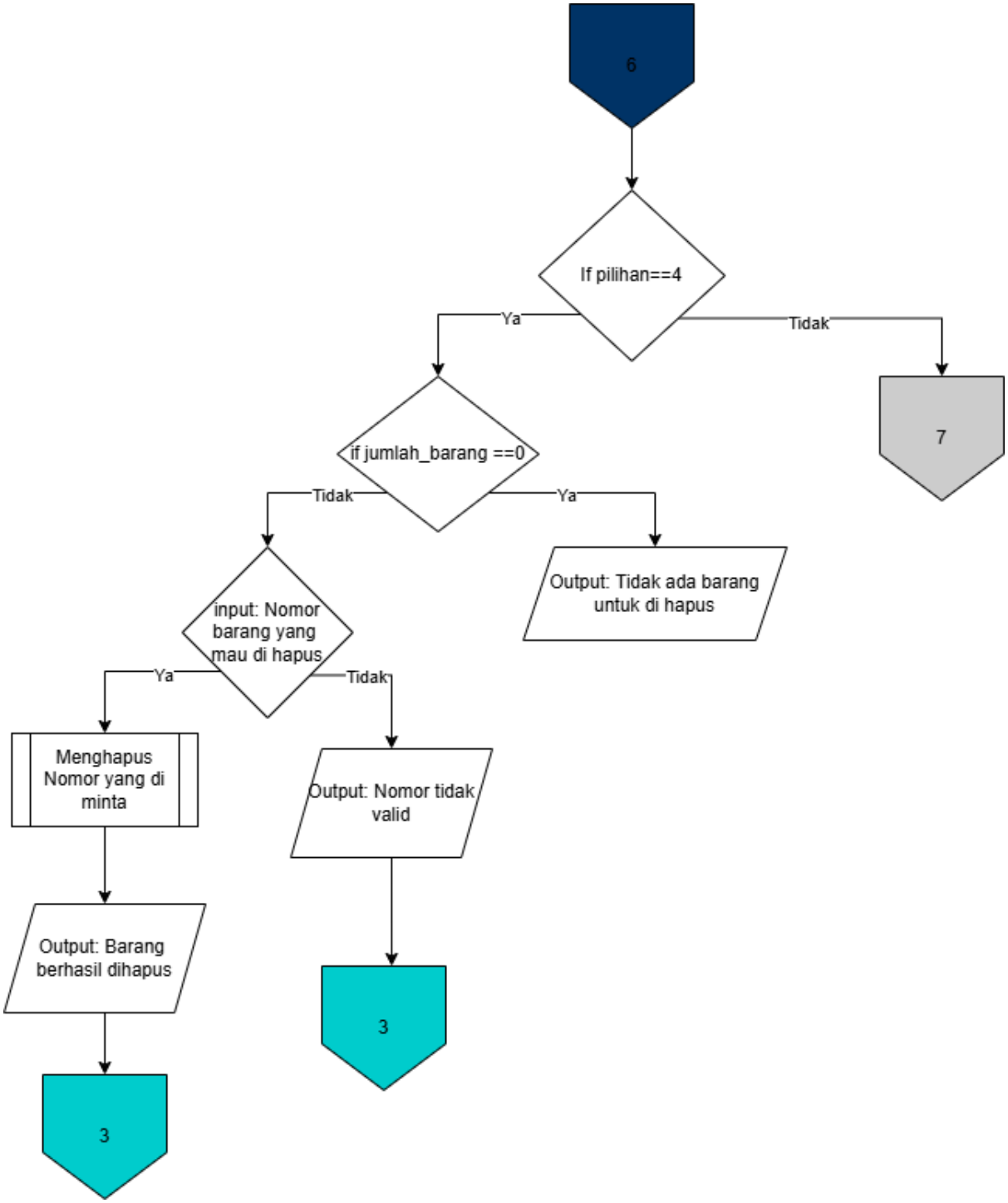

START
1
Output:
1.Login
2. Register
3. Keluar
1
if opsi 1
Ya
Tidak
3
input username
dan
Password
if opsi 2
Ya
Tidak
2
if opsi 3
Tidak
Ya
if username==reja dan
password==2409106107
Ya
Tidak
output:
pilihan tidak valid
Output: Keluar
dari program
2
3
Memeriksa
apakah user
terdaftar dalam
array
1
End
if username dan
password benar
Output: Login
Berhasil
Output: Login gagal!
coba lagi
U
if gagal <3
Ya
Tidak
Output: Kesempatan
login habis.Program
berhenti
3
2

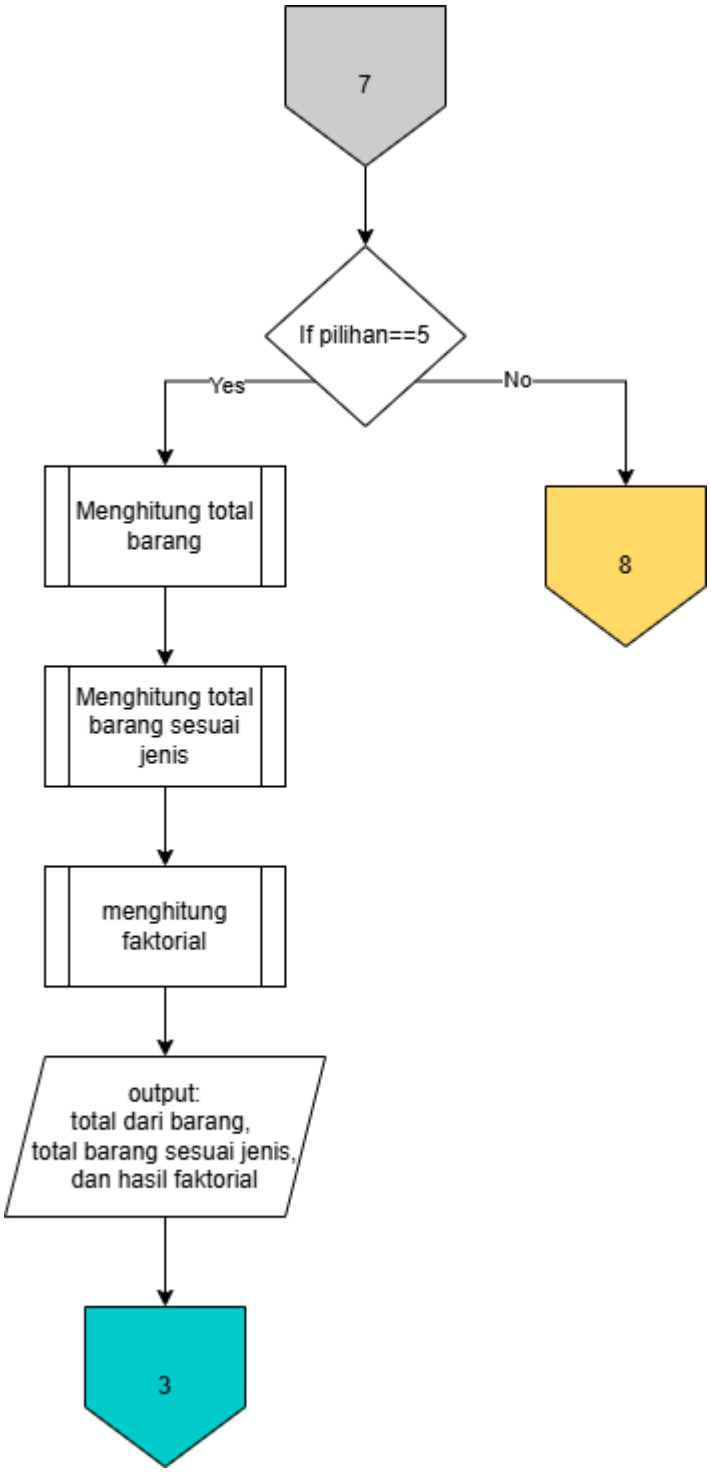

2
output: Masukan Username dan Password
menyimpan ke array
if jumlah user >= max user
Ya
Tidak
output: Kapasitas Penuh
Output: Registrasi Berhasil
1
1

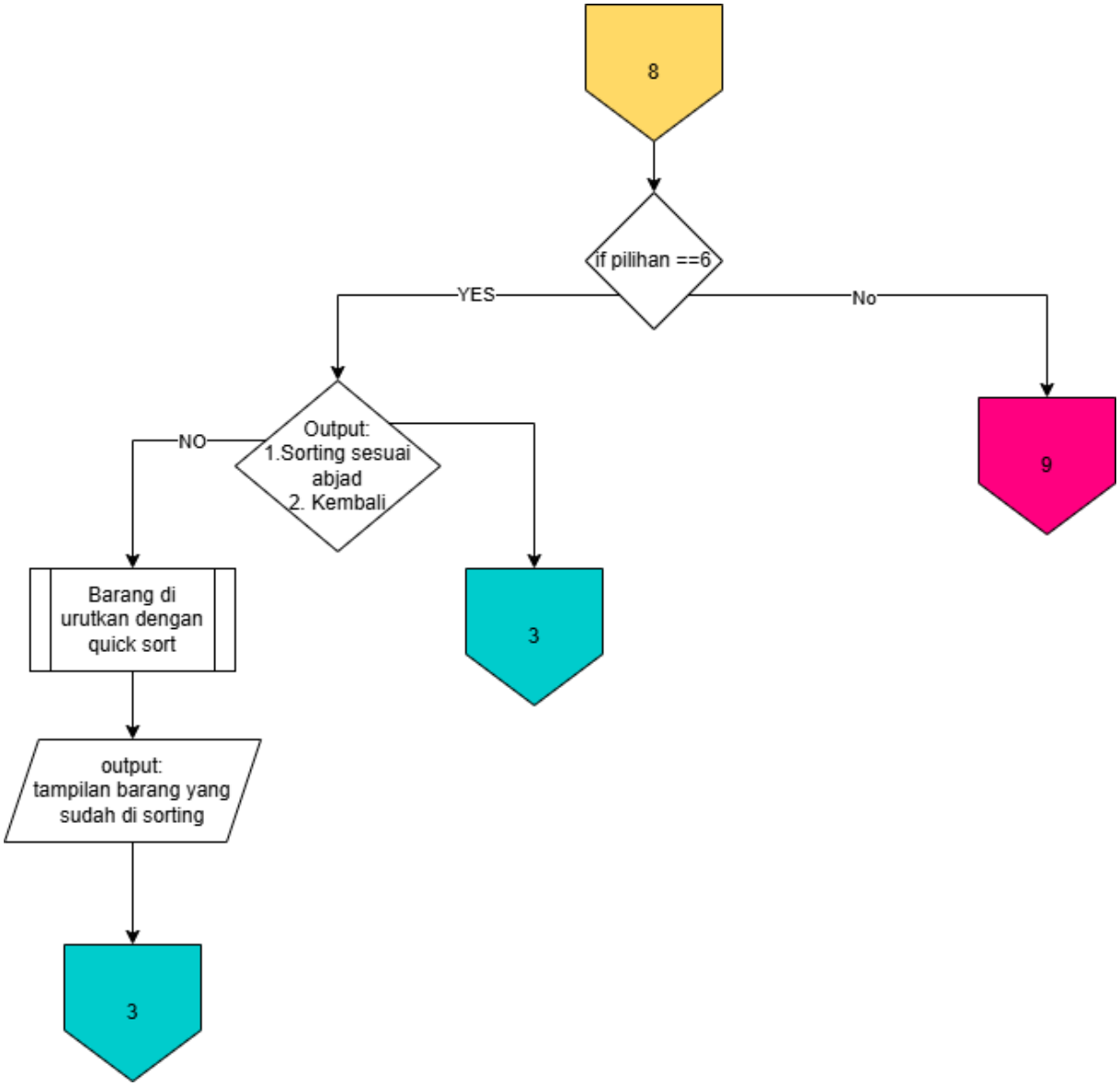

3
A
Output:Menu Vapestore
1.Tampilkan Barang
2.Tambah Barang
3.Ubah Barang
4.Hapus Barang
5. Info Barang
6. Sorting barang
7. Logout
if pilihan==1
Ya
Tidak
4
if barang ada
Tidak
Ya
Output: Belum ada barang dalam daftar
Output: Menampilkan Array data
A
A

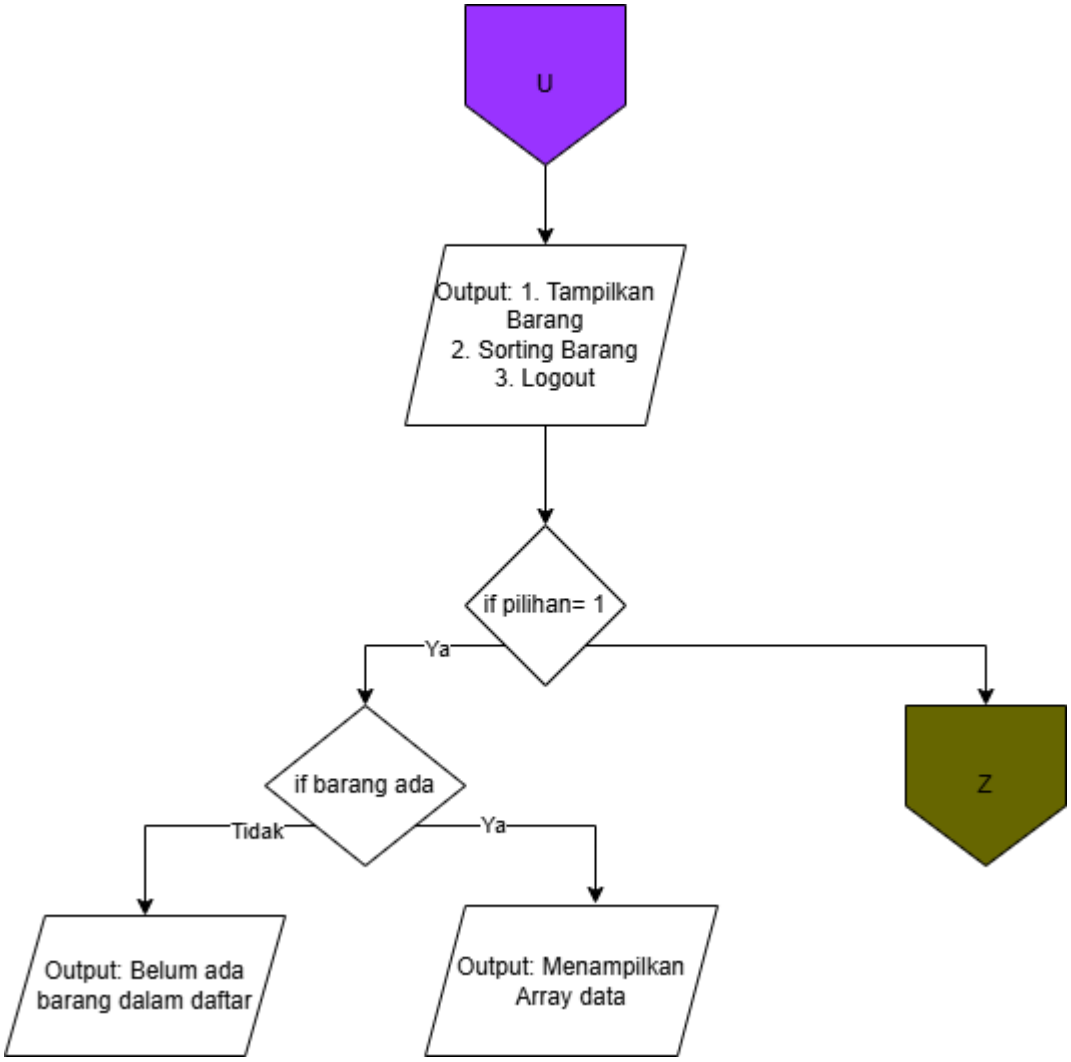

4
if pilihan==2
Ya
Tidak
5
if
jumlah_barang
>= 100
Tidak
Ya
Input:Nama barang
Jenis barang
Jumlah barang
Output: kapasitas
Penyimpanan
Penuh
Menyimpan ke array
3
Output: Barang
Berhasil di
tambahkan!
3

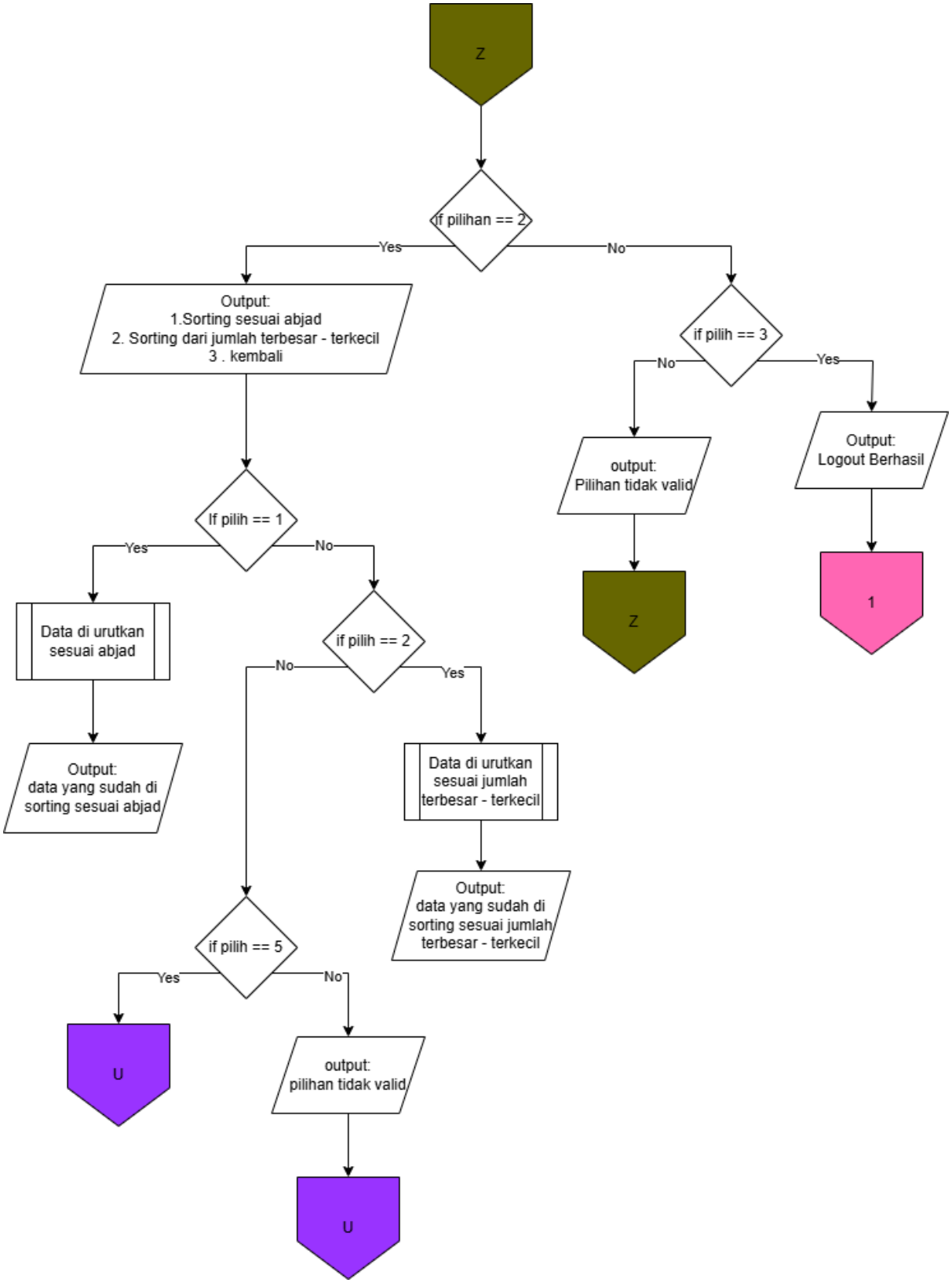

5
if pilihan== 3
Ya
Tidak
6
if jumlah barang==0
Tidak
Ya
input: Nomor barang yang mau di ubah
Output: Tidak ada barang yang diubah
Ya
Tidak
3
Input:Nama baru, jenis baru, jumlah baru
Output: Nomor tidak valid
Output: Barang berhasil diubah
3
3

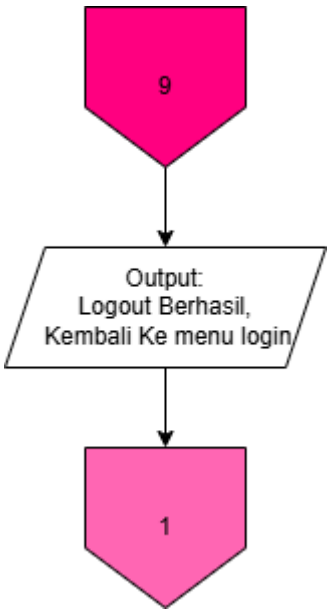

6
If pilihan==4
Ya
Tidak
7
if jumlah_barang ==0
Tidak
Ya
Output: Tidak ada barang untuk di hapus
input: Nomor barang yang mau di hapus
Ya
Tidak
Menghapus Nomor yang di minta
Output: Nomor tidak valid
Output: Barang berhasil dihapus
3
3


7
If pilihan==5
Yes
No
Menghitung total barang
8
Menghitung total barang sesuai jenis
menghitung faktorial
output:
total dari barang,
total barang sesuai jenis,
dan hasil faktorial
3


8
if pilihan ==6
YES
No
Output:
1.Sorting sesuai abjad
2. Kembali
NO
9
Barang di urutkan dengan quick sort
3
output:
tampilan barang yang sudah di sorting
3


U
Output: 1. Tampilkan Barang
2. Sorting Barang
3. Logout
if pilihan= 1
Ya
Z
if barang ada
Tidak
Ya
Output: Belum ada barang dalam daftar
Output: Menampilkan Array data


Z
if pilihan == 2
Yes
No
Output:
1.Sorting sesuai abjad
2. Sorting dari jumlah terbesar - terkecil
3 . kembali
if pilih == 3
No
Yes
output:
Pilihan tidak valid
Output:
Logout Berhasil
Z
1
If pilih == 1
Yes
No
Data di urutkan
sesuai abjad
if pilih == 2
No
Yes
Output:
data yang sudah di
sorting sesuai abjad
Data di urutkan
sesuai jumlah
terbesar - terkecil
Output:
data yang sudah di
sorting sesuai jumlah
terbesar - terkecil
if pilih == 5
Yes
No
U
output:
pilihan tidak valid
U


9
Output:
Logout Berhasil,
Kembali Ke menu login
1

2. Analisis Program

2.2 Tujuan dan Fungsi Program

1. Login & Registrasi

A. Tujuan:

- Mengamankan akses sistem dengan autentikasi berbasis pointer
- Memisahkan hak akses admin/user melalui pointer boolean
- Membatasi percobaan login (maksimal 3 kali)

B. Fungsi:

- Registrasi:
  - Menyimpan data user langsung ke memori via pointer VapeStore*
  - Memeriksa kapasitas maksimum user sebelum registrasi
- Login:
  - Verifikasi kredensial dengan akses pointer ke array users
  - Pemeriksaan khusus untuk admin (username == "reja")
  - Menggunakan dereference pointer (*loginBerhasil, *isAdmin) untuk kontrol akses
- Keamanan:
  - Auto-exit program setelah 3x percobaan login gagal

2. Fitur Admin

A. Tujuan:

- Manajemen stok efisien dengan operasi memori langsung
- Penyediaan tools sorting khusus (Quick Sort)

B. Fungsi:

- Tambah Barang:
  - Input data langsung ke array via store->barang[]
  - Pemeriksaan kapasitas maksimum barang
- Ubah Barang:
  - Modifikasi data via pointer ke elemen array (Barang*)
  - Menampilkan daftar barang sebelum pengeditan
- Hapus Barang:
  - Geser elemen array di memori dengan pointer arithmetic
  - Penyesuaian jumlahBarang setelah penghapusan
- Info Stok:
  - Hitung total via pointer tanpa duplikasi data
  - Menghitung berdasarkan jenis (Device/Liquid) dengan pointer ke string
  - Demo fungsi rekursif (faktorial)
- Sorting Barang:
  - Quick Sort ascending untuk pengurutan nama barang (A-Z)
  - Operasi dilakukan pada array temporary tanpa mengubah data asli

3. Fitur User

A. Tujuan:

- Akses read-only ke data dengan pointer aman
- Kemampuan sorting terbatas tanpa mengubah data asli

B. Fungsi:

- Tampilkan Barang:
  - Akses data via store->barang tanpa risiko modifikasi
  - Format tabel rapi dengan pointer ke struct Barang
- Sorting Barang:
  - Bubble Sort untuk pengurutan ascending nama (A-Z)
  - Selection Sort untuk pengurutan descending jumlah
  - Operasi dilakukan pada array temporary (salinan data)

2.3 Manfaat Utama

1. Optimasi Memori
   - Passing struct besar sebagai pointer (4 byte) bukan salinan
   - Operasi array langsung di memori (tambah/ubah/hapus)
   - Penggunaan array temporary untuk sorting tanpa modifikasi data original
2. Keamanan
   - Variabel login (*loginBerhasil, *isAdmin) dimodifikasi via pointer
   - Batas maksimum array dicek via store->jumlahBarang dan store->jumlahUser
   - Pembatasan akses modifikasi data untuk user biasa
3. Performa
   - Perhitungan stok cepat dengan pointer arithmetic
   - Komparasi string via dereference (*jenis)
   - Algoritma sorting optimal:
     - Quick Sort untuk admin
     - Bubble/Selection Sort untuk user
4. Keterbacaan Kode
   - Fungsi menerima parameter eksplisit (VapeStore*, Barang*)
   - Operasi CRUD terpisah jelas (admin vs user)
   - Pembagian logika sorting:
     - User: Bubble Sort (ascending) + Selection Sort (descending)
     - Admin: Quick Sort (ascending)
5. Antarmuka Pengguna
   - Tampilan tabel terformat rapi dengan pointer ke struct
   - Menu interaktif dengan validasi input
   - Pesan error spesifik untuk kondisi batas

3. Source Code

```
#include <iostream>
#include <iomanip>
#include <string>
#include <algorithm>
using namespace std;

int MAX_BARANG = 100;
int MAX_USER = 100;

struct Barang {
    string nama;
    string jenis;
    int jumlah;
};

struct User {
    string username;
    string password;
};

struct VapeStore {
    Barang barang[100];
    User users[100];
    int jumlahBarang = 6;
    int jumlahUser = 0;
};

void tampilkanMenuLogin();
void prosesLogin(VapeStore* store, bool* loginBerhasil, bool* isAdmin);
void prosesRegister(VapeStore* store);
void tampilkanMenuUser();
void tampilkanMenuAdmin();
void tampilkanBarang(VapeStore* store);
void tambahBarang(VapeStore* store);
void ubahBarangDenganPointer(Barang* barang, int jumlahBarang);
void hapusBarang(VapeStore* store);
void tampilkanHeaderTabel();
void tampilkanBarisBarang(Barang* barang, int index);
void tampilkanFooterTabel();
int hitungTotalBarang(VapeStore* store);
int hitungTotalBarangByJenis(VapeStore* store, string* jenis);
int faktorial(int n);
void tampilkanInfoTotalBarang(VapeStore* store);

// Sorting functions
void bubbleSortAscendingHuruf(Barang arr[], int n);
```

```
void selectionSortDescendingAngka(Barang arr[], int n);
void quickSortAscendingHuruf(Barang arr[], int low, int high);
int partition(Barang arr[], int low, int high);
void menuSortingUser(VapeStore* store);
void menuSortingAdmin(VapeStore* store);

// Fungsi utama
int main() {
    VapeStore store = {
        {{"Oxva xlim GO", "Device", 10}, {"Voopoo Drag X", "Device", 7},
{"TRML T99", "Device", 5},
         {"Makna V2 3mg", "Liquid", 15}, {"Bolu Lapis Talas V1 6mg",
"Liquid", 12}, {"The Orama V1 3mg", "Liquid", 20}},
        {},
        6,
        0
    };

    while (true) {
        tampilkanMenuLogin();
        int menuLogin;
        cout << "Pilihan: ";
        cin >> menuLogin;

        switch (menuLogin) {
            case 1: {
                bool loginBerhasil = false;
                bool isAdmin = false;
                prosesLogin(&store, &loginBerhasil, &isAdmin);

                if (loginBerhasil) {
                    if (isAdmin) {
                        // Menu Admin
                        while (true) {
                            tampilkanMenuAdmin();
                            int pilihan;
                            cout << "Pilihan: ";
                            cin >> pilihan;

                            switch (pilihan) {
                                case 1: tampilkanBarang(&store); break;
                                case 2: tambahBarang(&store); break;
                                case 3:
ubahBarangDenganPointer(store.barang, store.jumlahBarang); break;
                                case 4: hapusBarang(&store); break;
                                case 5: tampilkanInfoTotalBarang(&store);
break;
                                case 6: menuSortingAdmin(&store); break;
```

```
                            case 7:
                                cout << "Logout berhasil.\n";
                                goto logout;
                            default:
                                cout << "Pilihan tidak valid!\n";
                        }
                    }
                } else {
                    // Menu User Biasa
                    while (true) {
                        tampilkanMenuUser();
                        int pilihan;
                        cout << "Pilihan: ";
                        cin >> pilihan;

                        switch (pilihan) {
                            case 1: tampilkanBarang(&store); break;
                            case 2: menuSortingUser(&store); break;
                            case 3:
                                cout << "Logout berhasil.\n";
                                goto logout;
                            default:
                                cout << "Pilihan tidak valid!\n";
                        }
                    }
                }
                logout:;
            }
            break;
        }
        case 2:
            prosesRegister(&store);
            break;
        case 3:
            cout << "Keluar dari program.\n";
            return 0;
        default:
            cout << "Pilihan tidak valid!\n";
    }
}

    return 0;
}

void tampilkanMenuLogin() {
    cout << "\n=== Menu Login ===\n";
    cout << "1. Login\n";
    cout << "2. Register\n";
```

```
    cout << "3. Keluar\n";
}

void prosesLogin(VapeStore* store, bool* loginBerhasil, bool* isAdmin) {
    string username, password;
    int attempts = 0;
    *loginBerhasil = false;
    *isAdmin = false;

    while (attempts < 3 && !(*loginBerhasil)) {
        cout << "Username: ";
        cin >> username;
        cout << "Password: ";
        cin >> password;

        // Cek admin
        if (username == "reja" && password == "2409106107") {
            *loginBerhasil = true;
            *isAdmin = true;
            cout << "Login sebagai admin berhasil!\n";
            return;
        }

        // Cek user biasa
        for (int i = 0; i < store->jumlahUser; i++) {
            if (store->users[i].username == username && store-
>users[i].password == password) {
                *loginBerhasil = true;
                cout << "Login berhasil!\n";
                return;
            }
        }

        cout << "Login gagal! Coba lagi.\n";
        attempts++;
    }

    if (attempts == 3) {
        cout << "Kesempatan login habis. Program berhenti.\n";
        exit(0);
    }
}

void prosesRegister(VapeStore* store) {
    if (store->jumlahUser >= MAX_USER) {
        cout << "Kapasitas user penuh!\n";
        return;
    }
```

```
    cout << "Username: ";
    cin >> store->users[store->jumlahUser].username;
    cout << "Password: ";
    cin >> store->users[store->jumlahUser].password;
    store->jumlahUser++;
    cout << "Registrasi berhasil!\n";
}

void tampilkanMenuUser() {
    cout << "\n=== Menu User ===\n";
    cout << "1. Tampilkan Barang\n";
    cout << "2. Sorting Barang\n";
    cout << "3. Logout\n";
}

void tampilkanMenuAdmin() {
    cout << "\n=== Menu Admin ===\n";
    cout << "1. Tampilkan Barang\n";
    cout << "2. Tambah Barang\n";
    cout << "3. Ubah Barang\n";
    cout << "4. Hapus Barang\n";
    cout << "5. Info Total Barang\n";
    cout << "6. Sorting Barang\n";
    cout << "7. Logout\n";
}

void tampilkanBarang(VapeStore* store) {
    if (store->jumlahBarang == 0) {
        cout << "Belum ada barang dalam daftar.\n";
        return;
    }

    tampilkanHeaderTabel();
    for (int i = 0; i < store->jumlahBarang; i++) {
        tampilkanBarisBarang(&(store->barang[i]), i + 1);
    }
    tampilkanFooterTabel();
}

void tambahBarang(VapeStore* store) {
    if (store->jumlahBarang >= MAX_BARANG) {
        cout << "Kapasitas penyimpanan penuh!\n";
        return;
    }

    cout << "Nama barang: ";
    cin.ignore();
```

```
    getline(cin, store->barang[store->jumlahBarang].nama);
    cout << "Jenis (Device/Liquid): ";
    getline(cin, store->barang[store->jumlahBarang].jenis);
    cout << "Jumlah: ";
    cin >> store->barang[store->jumlahBarang].jumlah;
    store->jumlahBarang++;
    cout << "Barang berhasil ditambahkan!\n";
}

void ubahBarangDenganPointer(Barang* barang, int jumlahBarang) {
    if (jumlahBarang == 0) {
        cout << "Tidak ada barang untuk diubah.\n";
        return;
    }

    tampilkanHeaderTabel();
    for (int i = 0; i < jumlahBarang; i++) {
        tampilkanBarisBarang(&barang[i], i + 1);
    }
    tampilkanFooterTabel();

    int index;
    cout << "Pilih nomor barang yang ingin diubah: ";
    cin >> index;

    if (index < 1 || index > jumlahBarang) {
        cout << "Nomor tidak valid!\n";
        return;
    }

    Barang* barangToEdit = &barang[index - 1];
    cin.ignore();
    cout << "Nama baru: ";
    getline(cin, barangToEdit->nama);
    cout << "Jenis baru (Device/Liquid): ";
    getline(cin, barangToEdit->jenis);
    cout << "Jumlah baru: ";
    cin >> barangToEdit->jumlah;
    cout << "Barang berhasil diubah!\n";
}

void hapusBarang(VapeStore* store) {
    if (store->jumlahBarang == 0) {
        cout << "Tidak ada barang untuk dihapus.\n";
        return;
    }

    tampilkanBarang(store);
```

```
    int index;
    cout << "Pilih nomor barang yang akan dihapus: ";
    cin >> index;

    if (index < 1 || index > store->jumlahBarang) {
        cout << "Nomor tidak valid!\n";
        return;
    }

    for (int i = index - 1; i < store->jumlahBarang - 1; i++) {
        store->barang[i] = store->barang[i + 1];
    }
    store->jumlahBarang--;
    cout << "Barang berhasil dihapus!\n";
}

void tampilkanHeaderTabel() {
    cout << "+-----+----------------------+------------+--------+\n";
    cout << "| No  | Nama Barang          | Jenis      | Jumlah |\n";
    cout << "+-----+----------------------+------------+--------+\n";
}

void tampilkanBarisBarang(Barang* barang, int index) {
    cout << "| " << setw(3) << index << " | " << setw(20) << barang->nama <<
" | "
         << setw(10) << barang->jenis << " | "
         << setw(6) << barang->jumlah << " |\n";
}

void tampilkanFooterTabel() {
    cout << "+-----+----------------------+------------+--------+\n";
}

int hitungTotalBarang(VapeStore* store) {
    int total = 0;
    for (int i = 0; i < store->jumlahBarang; i++) {
        total += store->barang[i].jumlah;
    }
    return total;
}

int hitungTotalBarangByJenis(VapeStore* store, string* jenis) {
    int total = 0;
    for (int i = 0; i < store->jumlahBarang; i++) {
        if (store->barang[i].jenis == *jenis) {
            total += store->barang[i].jumlah;
        }
    }
```

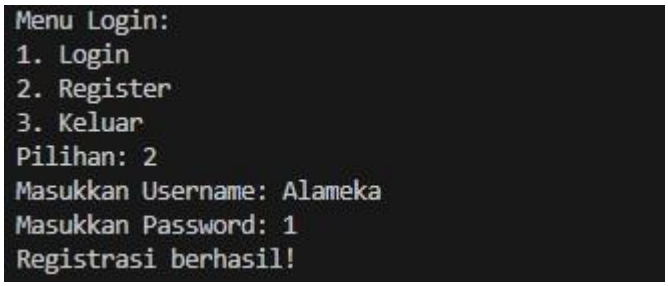
```
    return total;
}

int faktorial(int n) {
    if (n <= 1) return 1;
    return n * faktorial(n - 1);
}

void tampilkanInfoTotalBarang(VapeStore* store) {
    cout << "\n=== Informasi Total Barang ===\n";
    cout << "Total semua barang: " << hitungTotalBarang(store) << endl;

    string jenis1 = "Device";
    string jenis2 = "Liquid";
    cout << "Total Device: " << hitungTotalBarangByJenis(store, &jenis1) <<
endl;
    cout << "Total Liquid: " << hitungTotalBarangByJenis(store, &jenis2) <<
endl;

    cout << "\nDemo Fungsi Rekursif (Faktorial):\n";
    for (int i = 1; i <= 5; i++) {
        cout << "Faktorial " << i << ": " << faktorial(i) << endl;
    }
}

// Sorting functions implementation
void bubbleSortAscendingHuruf(Barang arr[], int n) {
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++) {
            if (arr[j].nama > arr[j+1].nama) {
                swap(arr[j], arr[j+1]);
            }
        }
    }
}

void selectionSortAscendingHuruf(Barang arr[], int n) {
    for (int i = 0; i < n-1; i++) {
        int min_idx = i;
        for (int j = i+1; j < n; j++) {
            if (arr[j].nama < arr[min_idx].nama) {
                min_idx = j;
            }
        }
        swap(arr[min_idx], arr[i]);
    }
}
```

```
void bubbleSortDescendingAngka(Barang arr[], int n) {
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++) {
            if (arr[j].jumlah < arr[j+1].jumlah) {
                swap(arr[j], arr[j+1]);
            }
        }
    }
}

void selectionSortDescendingAngka(Barang arr[], int n) {
    for (int i = 0; i < n-1; i++) {
        int max_idx = i;
        for (int j = i+1; j < n; j++) {
            if (arr[j].jumlah > arr[max_idx].jumlah) {
                max_idx = j;
            }
        }
        swap(arr[max_idx], arr[i]);
    }
}

int partition(Barang arr[], int low, int high) {
    string pivot = arr[high].nama;
    int i = (low - 1);

    for (int j = low; j <= high - 1; j++) {
        if (arr[j].nama < pivot) {
            i++;
            swap(arr[i], arr[j]);
        }
    }
    swap(arr[i + 1], arr[high]);
    return (i + 1);
}

void quickSortAscendingHuruf(Barang arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSortAscendingHuruf(arr, low, pi - 1);
        quickSortAscendingHuruf(arr, pi + 1, high);
    }
}

void menuSortingUser(VapeStore* store) {
    if (store->jumlahBarang == 0) {
        cout << "Tidak ada barang untuk diurutkan.\n";
        return;
```

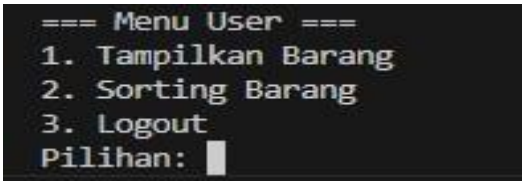
```
    }

    cout << "\n=== Menu Sorting User ===\n";
    cout << "1. Sorting Sesuai Abjad (A-Z)\n";
    cout << "2. Sorting dari jumlah terbesar - terkecil\n";
    cout << "3. Kembali\n";

    int pilihan;
    cout << "Pilihan: ";
    cin >> pilihan;

    // Buat salinan array untuk sorting
    Barang temp[100];
    for (int i = 0; i < store->jumlahBarang; i++) {
        temp[i] = store->barang[i];
    }

    switch (pilihan) {
        case 1:
        bubbleSortAscendingHuruf(temp, store->jumlahBarang);
        cout << "\nHasil Sorting sesuai abjad:\n";
        break;
        case 2:
        selectionSortDescendingAngka(temp, store->jumlahBarang);
        cout << "\nHasil Sorting dari jumlah angka terbesar ke terkecil:\n";
        break;
        case 3:
            return;
        default:
            cout << "Pilihan tidak valid!\n";
            return;
    }

    // Tampilkan hasil sorting
    tampilkanHeaderTabel();
    for (int i = 0; i < store->jumlahBarang; i++) {
        tampilkanBarisBarang(&temp[i], i + 1);
    }
    tampilkanFooterTabel();
}

void menuSortingAdmin(VapeStore* store) {
    if (store->jumlahBarang == 0) {
        cout << "Tidak ada barang untuk diurutkan.\n";
        return;
    }

    cout << "\n=== Menu Sorting Admin ===\n";
```

```
    cout << "1. Sorting Sesuai Abjad (A-Z):\n";
    cout << "2. Kembali\n";

    int pilihan;
    cout << "Pilihan: ";
    cin >> pilihan;

    // Buat salinan array untuk sorting
    Barang temp[100];
    for (int i = 0; i < store->jumlahBarang; i++) {
        temp[i] = store->barang[i];
    }

    switch (pilihan) {
        case 1:
            quickSortAscendingHuruf(temp, 0, store->jumlahBarang - 1);
            cout << "\nSorting Sesuai Abjad (A-Z):\n";
            break;
        case 2:
            return;
        default:
            cout << "Pilihan tidak valid!\n";
            return;
    }

    // Tampilkan hasil sorting
    tampilkanHeaderTabel();
    for (int i = 0; i < store->jumlahBarang; i++) {
        tampilkanBarisBarang(&temp[i], i + 1);
    }
    tampilkanFooterTabel();
}
```

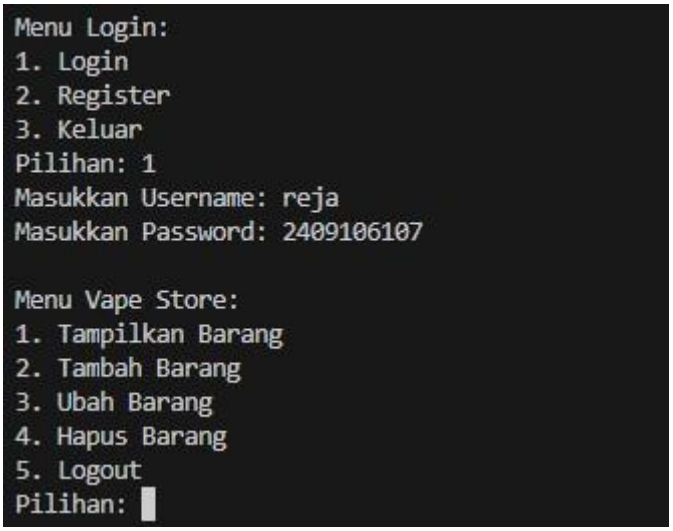
4. Uji Coba dan Hasil Output



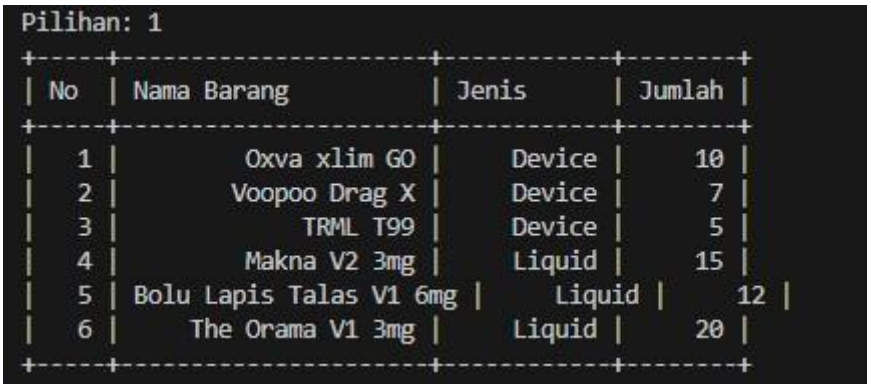
Gambar 4.1 Output Register



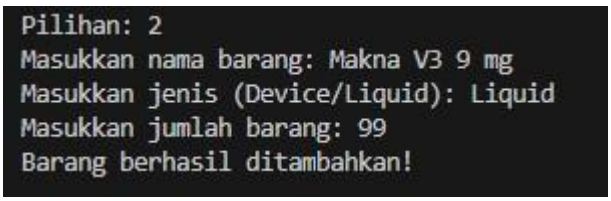
Gambar 4.2 Menu User



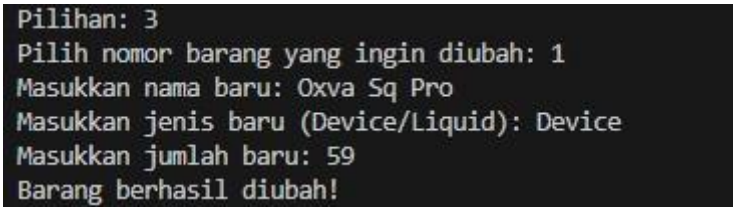
Gambar 4.3 Login dan Menu Admin



Gambar 4.4 Output Pilihan 1 Admin



Gambar 4.4 Output Pilihan 2



Gambar 4.5 Output Pilihan 3

```
Pilihan: 4
Masukkan nomor barang yang akan dihapus: 5
Barang berhasil dihapus!
```

Gambar 4.6 Output Pilihan 4

```
Pilihan: 6
Logout berhasil.
```

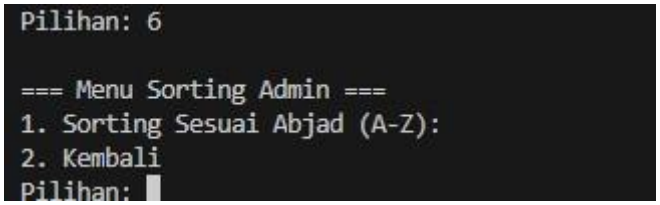
Gambar 4.7 Ouput Pilihan

```
Pilihan: 5

=== Informasi Total Barang ===
Total semua barang: 69
Total Device: 22
Total Liquid: 47

Demo Fungsi Rekursif (Faktorial):
Faktorial 1: 1
Faktorial 2: 2
Faktorial 3: 6
Faktorial 4: 24
Faktorial 5: 120
```

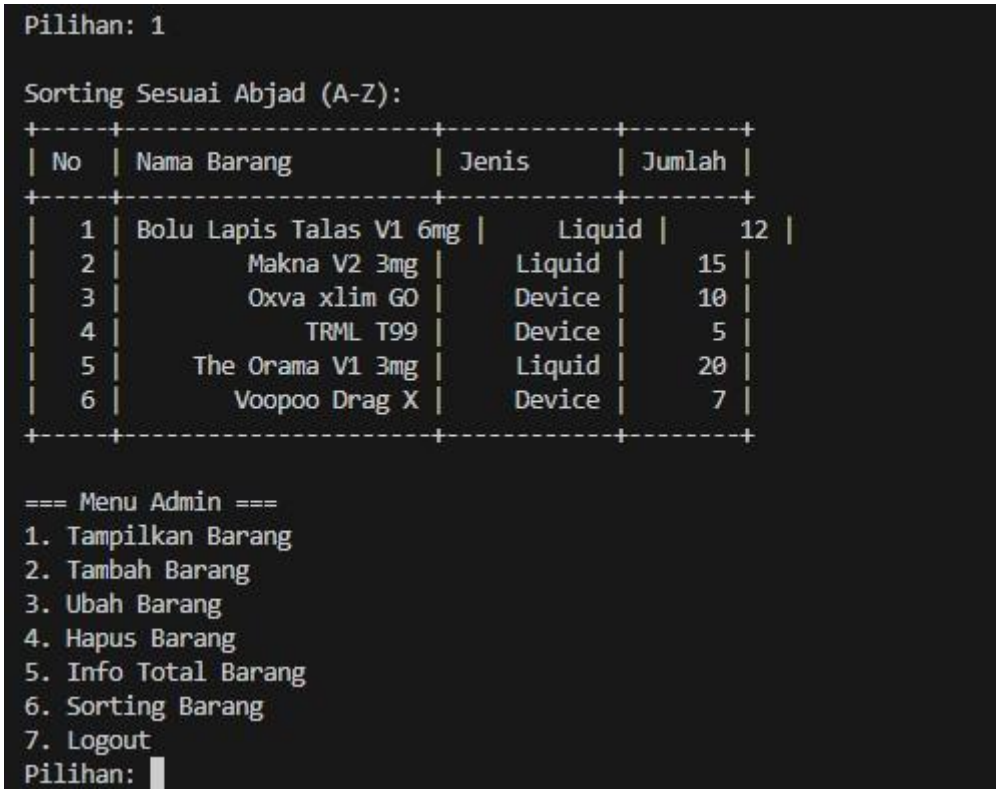
Gambar 4.8 Output Pilihan 1

```
Pilihan: 1
+-----+----------------------+------------+--------+
| No  | Nama Barang          | Jenis      | Jumlah |
+-----+----------------------+------------+--------+
|   1 |         Oxva xlim GO |     Device |     10 |
|   2 |        Voopoo Drag X |     Device |      7 |
|   3 |             TRML T99 |     Device |      5 |
|   4 |         Makna V2 3mg |     Liquid |     15 |
|   5 | Bolu Lapis Talas V1 6mg |     Liquid |     12 |
|   6 |     The Orama V1 3mg |     Liquid |     20 |
+-----+----------------------+------------+--------+
```

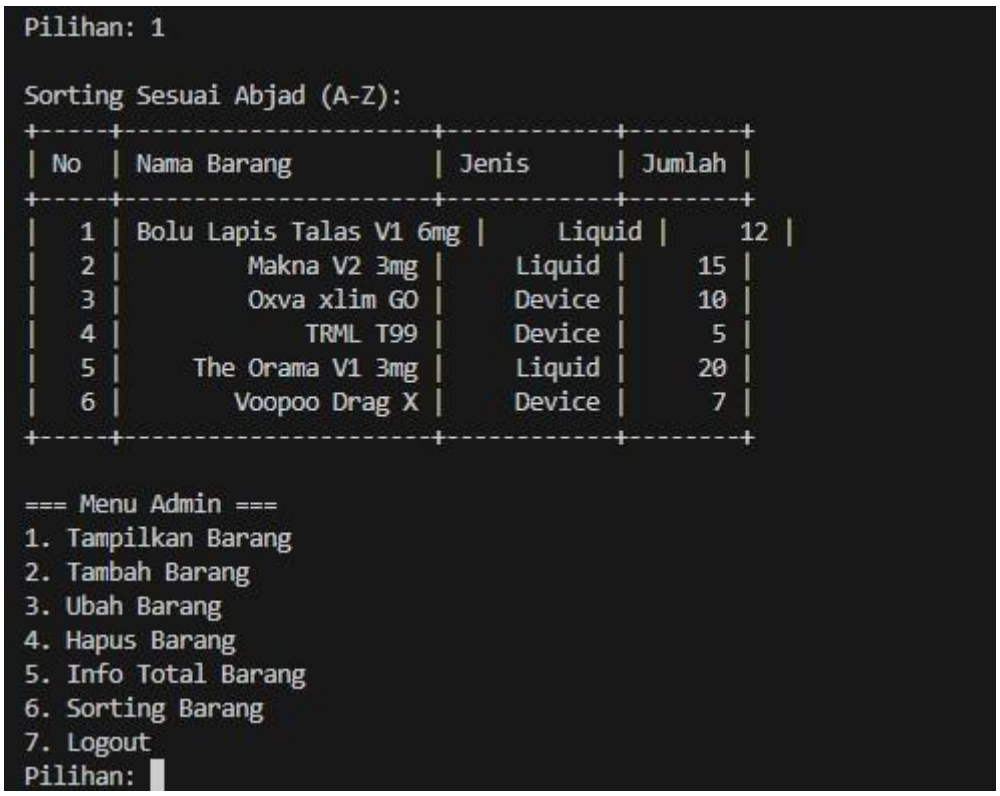
Gambar 4.9 Output Pilihan 5



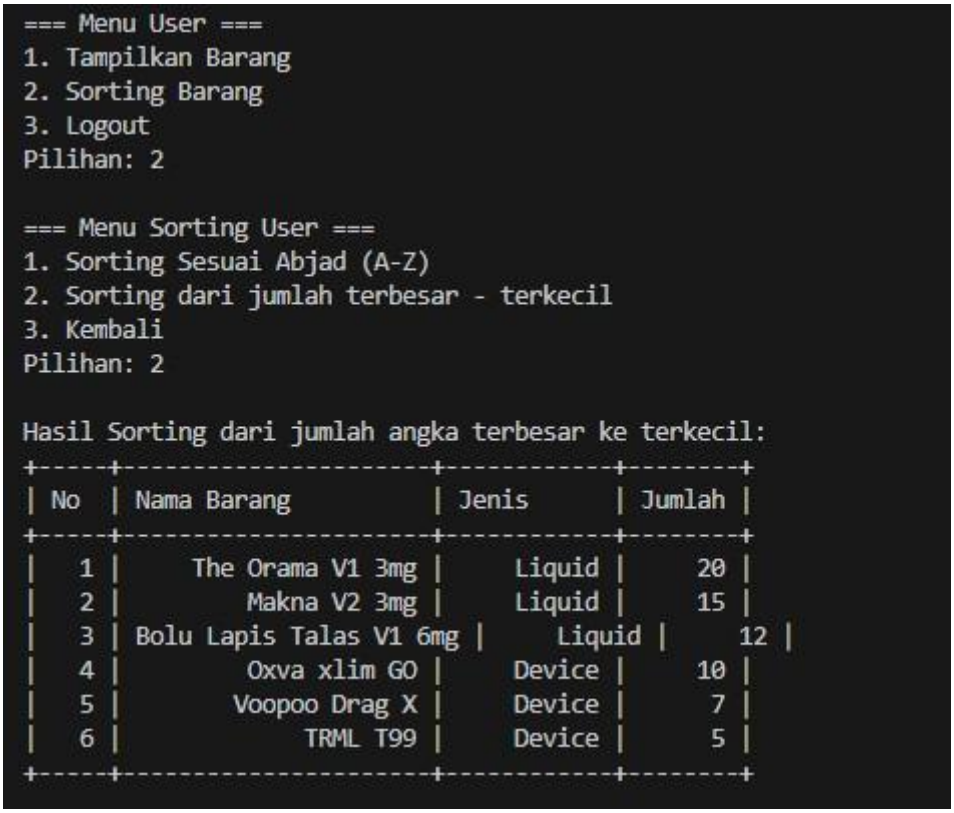
Gambar 4.10 Output pilihan 6



Gambar 4.11 Output pilihan 1 dalam sorting admin

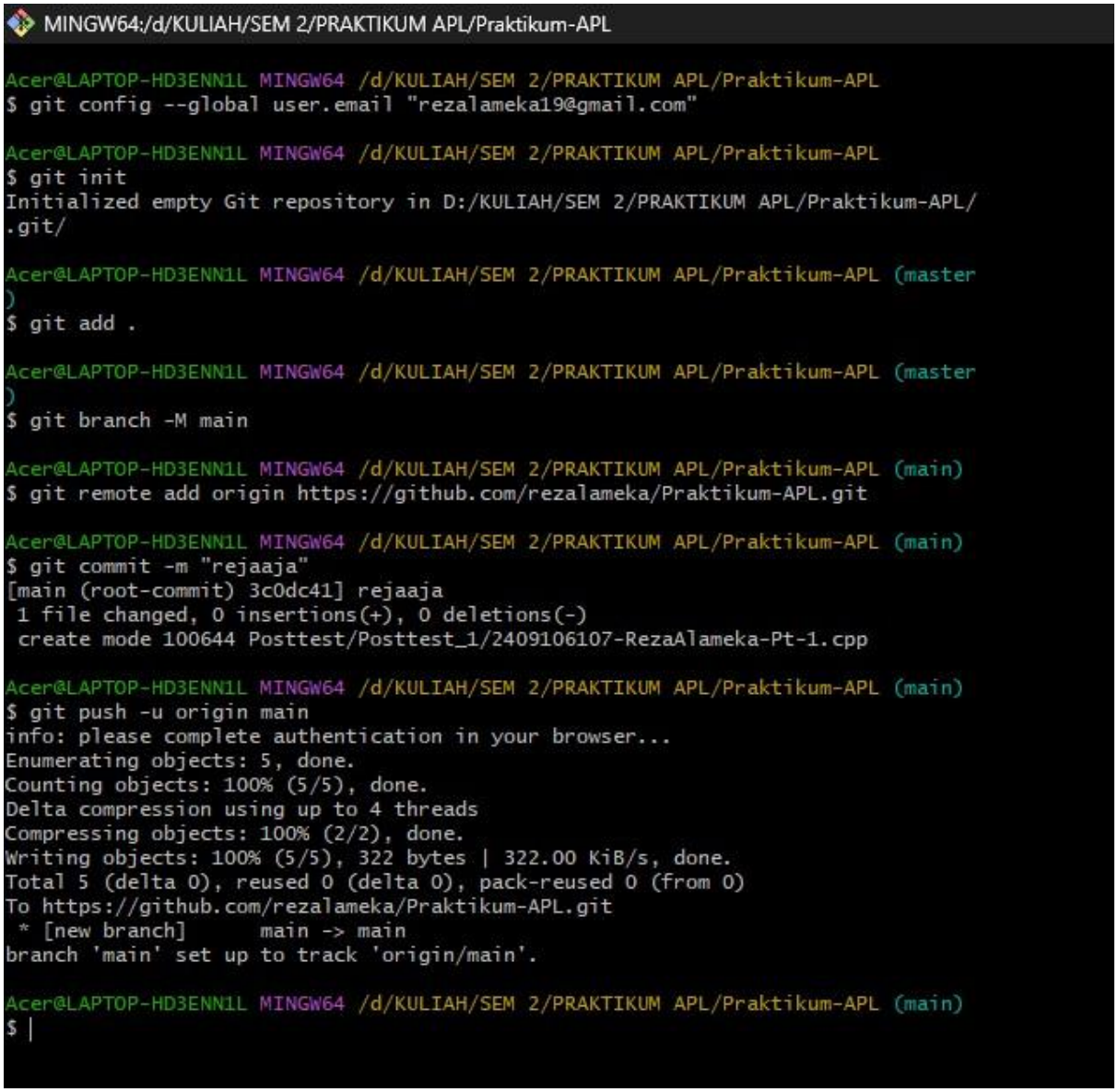


Gambar 4.12 Sorting sesuai abjad User



Gambar 4.13 Sorting user dari jumlah terbesar ke terkecil

5. Langkah-Langkah Git pada VSCode



1. Inisialisasi Git (git init): Membuat repository Git lokal.
2. Konfigurasi Git (git config): Menetapkan email pengguna.
3. Menambahkan file (git add .): Menambahkan semua file ke staging area.
4. Membuat branch utama (git branch -M main): Mengubah nama branch ke main.
5. Menambahkan remote repository (git remote add origin <URL>): Menghubungkan repository lokal ke GitHub.
6. Commit perubahan (git commit -m "rejaaja"): Menyimpan perubahan ke repository lokal.
7. Push ke GitHub (git push -u origin main): Mengunggah perubahan ke repository di GitHub