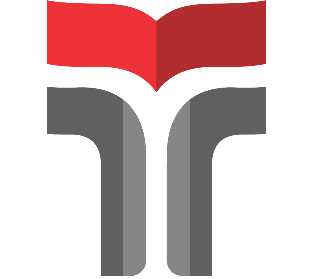
**LAPORAN PRAKTIKUM STRUKTUR DATA DAN ALGORITMA**

**MODUL IV**

**LINKED LIST CIRCULAR DAN NON CIRCULAR** 

# Disusun Oleh :

NAMA : Aulia Radix Putra Winarko  
NIM : 2311102056

# Dosen

Wahyu Andi Saputra, S.Pd., M.Eng

**PROGRAM STUDI S1 TEKNIK INFORMATIKA   
FAKULTAS INFORMATIKA**

**INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
2024**

1. **Dasar Teori**

**Linked List** adalah struktur data yang terdiri dari urutan elemen data yang disebut node. Setiap node terhubung satu sama lain dengan pointer yang menunjukkan alamat node selanjutnya.

## Linked List Non Circular

* **Struktur**
* Setiap node memiliki dua field: data dan pointer next.
* Pointer next pada node terakhir menunjuk ke NULL.
* Akses data hanya bisa dilakukan dari node pertama (head).
* **Operasi**
* Insertion:
* Di awal (head)
* Di tengah
* Di akhir
* Deletion
* Di awal (head)
* Di tengah
* Di akhir
* Traversal
* Dari awal (head) sampai akhir
* **Kelebihan**
* Implementasi mudah.
* Memori yang dialokasikan dinamis.
* Efisien untuk penyisipan dan penghapusan di tengah.
* **Kekurangan**
* Akses data hanya bisa dilakukan dari awal (head).
* Penghapusan node di tengah membutuhkan traversal.

## Linked List Circular

* **Struktur**
* Setiap node memiliki dua field: data dan pointer next.
* Pointer next pada node terakhir menunjuk ke node pertama (head).
* Akses data bisa dilakukan dari node manapun.
* **Operasi**
* Insertion:
* Di awal (head)
* Di tengah
* Di akhir
* Deletion
* Di awal (head)
* Di tengah
* Di akhir
* Traversal
* Bisa dimulai dari node manapun
* Berhenti ketika kembali ke node awal
* **Kelebihan**
* Akses data bisa dilakukan dari node manapun.
* Efisien untuk operasi insert dan delete di tengah.
* **Kekurangan**
* Implementasi sedikit lebih rumit.
* Penghapusan node di tengah membutuhkan traversal.

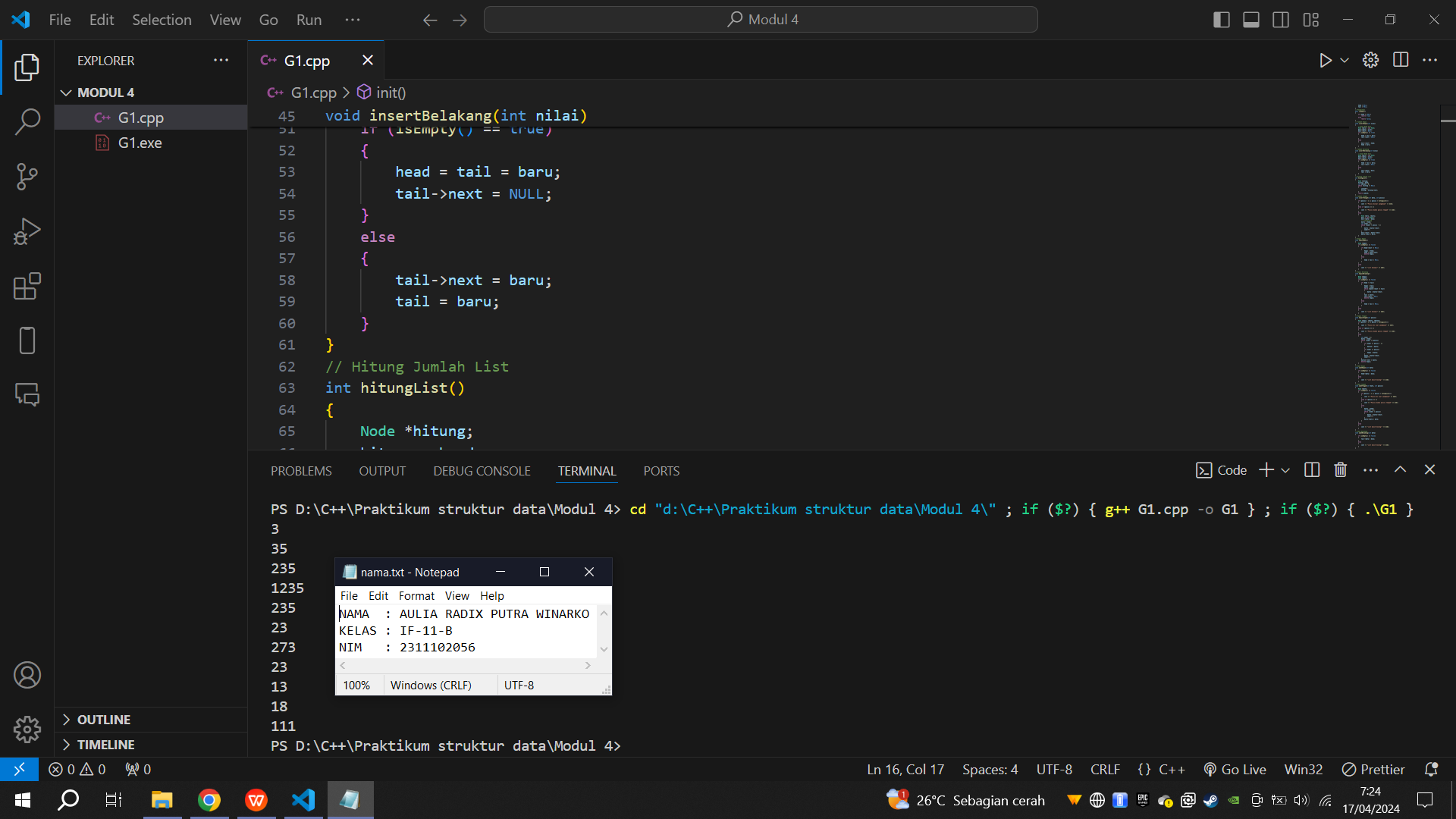
1. **Guided** [

Guided 1

**Linked List Non Circular**

|  |
| --- |
| #include <iostream>  using namespace std;  /// PROGRAM SINGLE LINKED LIST NON-CIRCULAR  // Deklarasi Struct Node  struct Node  {      int data;      Node \*next;  };  Node \*head;  Node \*tail;  // Inisialisasi Node  void init()  {      head = NULL;      tail = NULL;  }  // Pengecekan  bool isEmpty()  {      if (head == NULL)          return true;      else          return false;  }  // Tambah Depan  void insertDepan(int nilai)  {      // Buat Node baru      Node \*baru = new Node;      baru->data = nilai;      baru->next = NULL;      if (isEmpty() == true)      {          head = tail = baru;          tail->next = NULL;      }      else      {          baru->next = head;          head = baru;      }  }  // Tambah Belakang  void insertBelakang(int nilai)  {      // Buat Node baru      Node \*baru = new Node;      baru->data = nilai;      baru->next = NULL;      if (isEmpty() == true)      {          head = tail = baru;          tail->next = NULL;      }      else      {          tail->next = baru;          tail = baru;      }  }  // Hitung Jumlah List  int hitungList()  {      Node \*hitung;      hitung = head;      int jumlah = 0;      while (hitung != NULL)      {          jumlah++;          hitung = hitung->next;      }      return jumlah;  }  // Tambah Tengah  void insertTengah(int data, int posisi)  {      if (posisi < 1 || posisi > hitungList())      {          cout << "Posisi diluar jangkauan" << endl;      }      else if (posisi == 1)      {          cout << "Posisi bukan posisi tengah" << endl;      }      else      {          Node \*baru, \*bantu;          baru = new Node();          baru->data = data;          // tranversing          bantu = head;          int nomor = 1;          while (nomor < posisi - 1)          {              bantu = bantu->next;              nomor++;          }          baru->next = bantu->next;          bantu->next = baru;      }  }  // Hapus Depan  void hapusDepan()  {      Node \*hapus;      if (isEmpty() == false)      {          if (head->next != NULL)          {              hapus = head;              head = head->next;              delete hapus;          }          else          {              head = tail = NULL;          }      }      else      {          cout << "List kosong!" << endl;      }  }  // Hapus Belakang  void hapusBelakang()  {      Node \*hapus;      Node \*bantu;      if (isEmpty() == false)      {          if (head != tail)          {              hapus = tail;              bantu = head;              while (bantu->next != tail)              {                  bantu = bantu->next;              }              tail = bantu;              tail->next = NULL;              delete hapus;          }          else          {              head = tail = NULL;          }      }      else      {          cout << "List kosong!" << endl;      }  }  // Hapus Tengah  void hapusTengah(int posisi)  {      Node \*hapus, \*bantu, \*bantu2;      if (posisi < 1 || posisi > hitungList())      {          cout << "Posisi di luar jangkauan" << endl;      }      else if (posisi == 1)      {          cout << "Posisi bukan posisi tengah" << endl;      }      else      {          int nomor = 1;          bantu = head;          while (nomor <= posisi)          {              if (nomor == posisi - 1)              {                  bantu2 = bantu;              }              if (nomor == posisi)              {                  hapus = bantu;              }              bantu = bantu->next;              nomor++;          }          bantu2->next = bantu;          delete hapus;      }  }  // Ubah Depan  void ubahDepan(int data)  {      if (isEmpty() == false)      {          head->data = data;      }      else      {          cout << "List masih kosong!" << endl;      }  }  // Ubah Tengah  void ubahTengah(int data, int posisi)  {      Node \*bantu;      if (isEmpty() == false)      {          if (posisi < 1 || posisi > hitungList())          {              cout << "Posisi di luar jangkauan" << endl;          }          else if (posisi == 1)          {              cout << "Posisi bukan posisi tengah" << endl;          }          else          {              bantu = head;              int nomor = 1;              while (nomor < posisi)              {                  bantu = bantu->next;                  nomor++;              }              bantu->data = data;          }      }      else      {          cout << "List masih kosong!" << endl;      }  }  // Ubah Belakang  void ubahBelakang(int data)  {      if (isEmpty() == false)      {          tail->data = data;      }      else      {          cout << "List masih kosong!" << endl;      }  }  // Hapus List  void clearList()  {      Node \*bantu, \*hapus;      bantu = head;      while (bantu != NULL)      {          hapus = bantu;          bantu = bantu->next;          delete hapus;      }      head = tail = NULL;      cout << "List berhasil terhapus!" << endl;  }  // Tampilkan List  void tampil()  {      Node \*bantu;      bantu = head;      if (isEmpty() == false)      {          while (bantu != NULL)          {              cout << bantu->data << ends;              bantu = bantu->next;          }          cout << endl;      }      else      {          cout << "List masih kosong!" << endl;      }  }  int main()  {      init();      insertDepan(3);      tampil();      insertBelakang(5);      tampil();      insertDepan(2);      tampil();      insertDepan(1);      tampil();      hapusDepan();      tampil();      hapusBelakang();      tampil();      insertTengah(7, 2);      tampil();      hapusTengah(2);      tampil();      ubahDepan(1);      tampil();      ubahBelakang(8);      tampil();      ubahTengah(11, 2);      tampil();      return 0;  } |

Screenshots Output



Deskripsi:

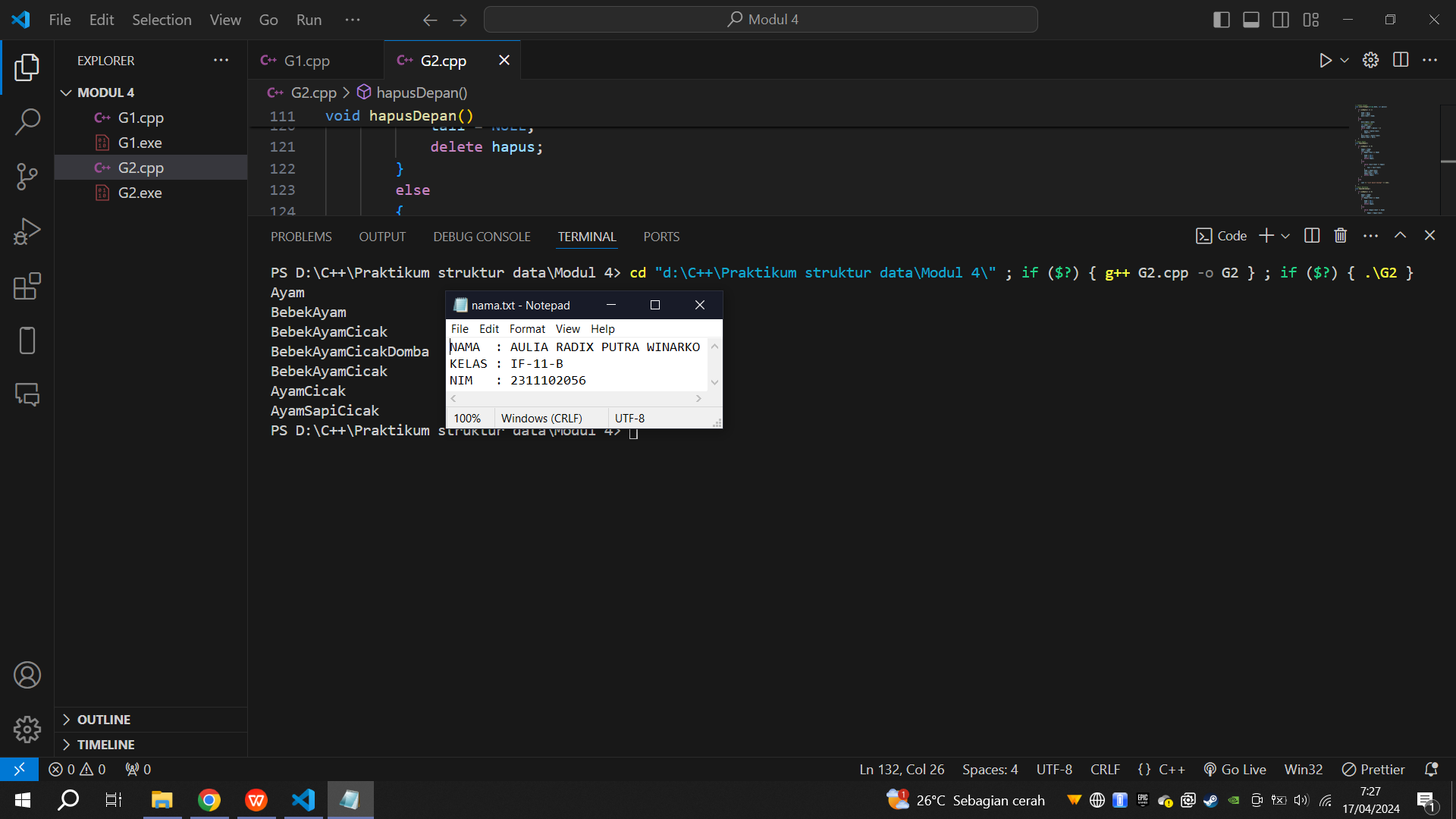
Kode C++ di atas adalah implementasi dari sebuah program yang menggunakan linked list untuk menyimpan dan mengelola data. Program ini menunjukkan cara menggunakan linked list untuk menyimpan data dan melakukan operasi seperti penambahan, penghapusan, dan pengubahan data pada linked list.

Guided 2

**Linked List Circular**

|  |
| --- |
| #include <iostream>  using namespace std;  /// PROGRAM SINGLE LINKED LIST CIRCULAR  // Deklarasi Struct Node  struct Node  {      string data;      Node \*next;  };  Node \*head, \*tail, \*baru, \*bantu, \*hapus;  void init()  {      head = NULL;      tail = head;  }  // Pengecekan  int isEmpty()  {      if (head == NULL)          return 1; // true      else          return 0; // false  }  // Buat Node Baru  void buatNode(string data)  {      baru = new Node;      baru->data = data;      baru->next = NULL;  }  // Hitung List  int hitungList()  {      bantu = head;      int jumlah = 0;      while (bantu != NULL)      {          jumlah++;          bantu = bantu->next;      }      return jumlah;  }  // Tambah Depan  void insertDepan(string data)  {      // Buat Node baru      buatNode(data);      if (isEmpty() == 1)      {          head = baru;          tail = head;          baru->next = head;      }      else      {          while (tail->next != head)          {              tail = tail->next;          }          baru->next = head;          head = baru;          tail->next = head;      }  }  // Tambah Belakang  void insertBelakang(string data)  {      // Buat Node baru      buatNode(data);      if (isEmpty() == 1)      {          head = baru;          tail = head;          baru->next = head;      }      else      {          while (tail->next != head)          {              tail = tail->next;          }          tail->next = baru;          baru->next = head;      }  }  // Tambah Tengah  void insertTengah(string data, int posisi)  {      if (isEmpty() == 1)      {          head = baru;          tail = head;          baru->next = head;      }      else      {          baru->data = data;          // transversing          int nomor = 1;          bantu = head;          while (nomor < posisi - 1)          {              bantu = bantu->next;              nomor++;          }          baru->next = bantu->next;          bantu->next = baru;      }  }  // Hapus Depan  void hapusDepan()  {      if (isEmpty() == 0)      {          hapus = head;          tail = head;          if (hapus->next == head)          {              head = NULL;              tail = NULL;              delete hapus;          }          else          {              while (tail->next != hapus)              {                  tail = tail->next;              }              head = head->next;              tail->next = head;              hapus->next = NULL;              delete hapus;          }      }      else      {          cout << "List masih kosong!" << endl;      }  }  // Hapus Belakang  void hapusBelakang()  {      if (isEmpty() == 0)      {          hapus = head;          tail = head;          if (hapus->next == head)          {              head = NULL;              tail = NULL;              delete hapus;          }          else          {              while (hapus->next != head)              {                  hapus = hapus->next;              }              while (tail->next != hapus)              {                  tail = tail->next;              }              tail->next = head;              hapus->next = NULL;              delete hapus;          }      }      else      {          cout << "List masih kosong!" << endl;      }  }  // Hapus Tengah  void hapusTengah(int posisi)  {      if (isEmpty() == 0)      {          // transversing          int nomor = 1;          bantu = head;          while (nomor < posisi - 1)          {              bantu = bantu->next;              nomor++;          }          hapus = bantu->next;          bantu->next = hapus->next;          delete hapus;      }      else      {          cout << "List masih kosong!" << endl;      }  }  // Hapus List  void clearList()  {      if (head != NULL)      {          hapus = head->next;          while (hapus != head)          {              bantu = hapus->next;              delete hapus;              hapus = bantu;          }          delete head;          head = NULL;      }      cout << "List berhasil terhapus!" << endl;  }  // Tampilkan List  void tampil()  {      if (isEmpty() == 0)      {          tail = head;          do          {              cout << tail->data << ends;              tail = tail->next;          } while (tail != head);          cout << endl;      }      else      {          cout << "List masih kosong!" << endl;      }  }  int main()  {      init();      insertDepan("Ayam");      tampil();      insertDepan("Bebek");      tampil();      insertBelakang("Cicak");      tampil();      insertBelakang("Domba");      tampil();      hapusBelakang();      tampil();      hapusDepan();      tampil();      insertTengah("Sapi", 2);      tampil();      hapusTengah(2);      tampil();      return 0;  } |

Screenshots Output



Deskripsi:

Kode C++ di atas adalah implementasi dari sebuah program yang menggunakan linked list untuk menyimpan dan mengelola data secara circular. Program ini menunjukkan cara menggunakan linked list secara circular untuk menyimpan data dan melakukan operasi seperti penambahan, penghapusan, dan penampilan data pada linked list. Dalam linked list circular, tail->next menunjuk kembali ke head, sehingga membentuk siklus tertutup

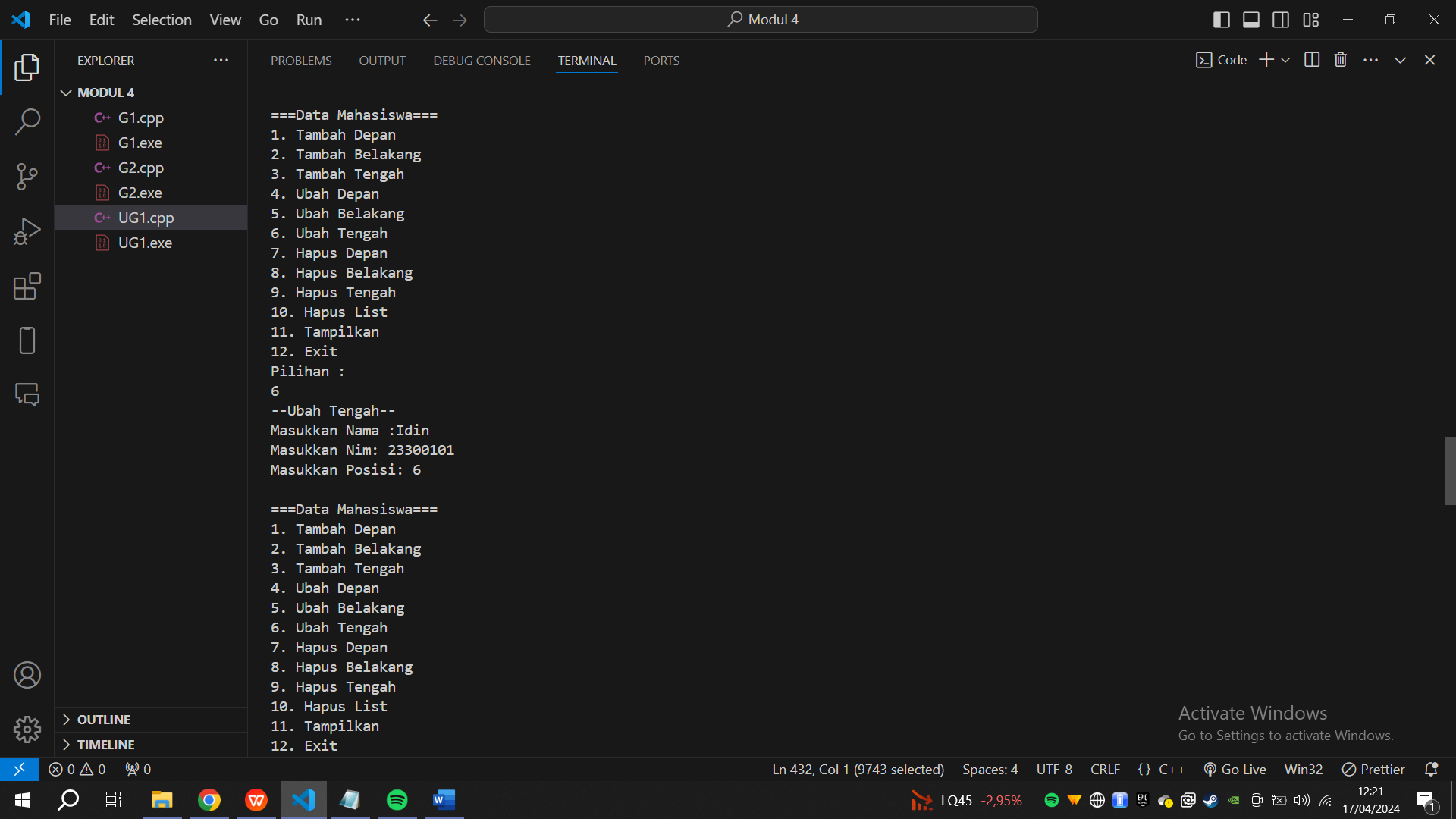
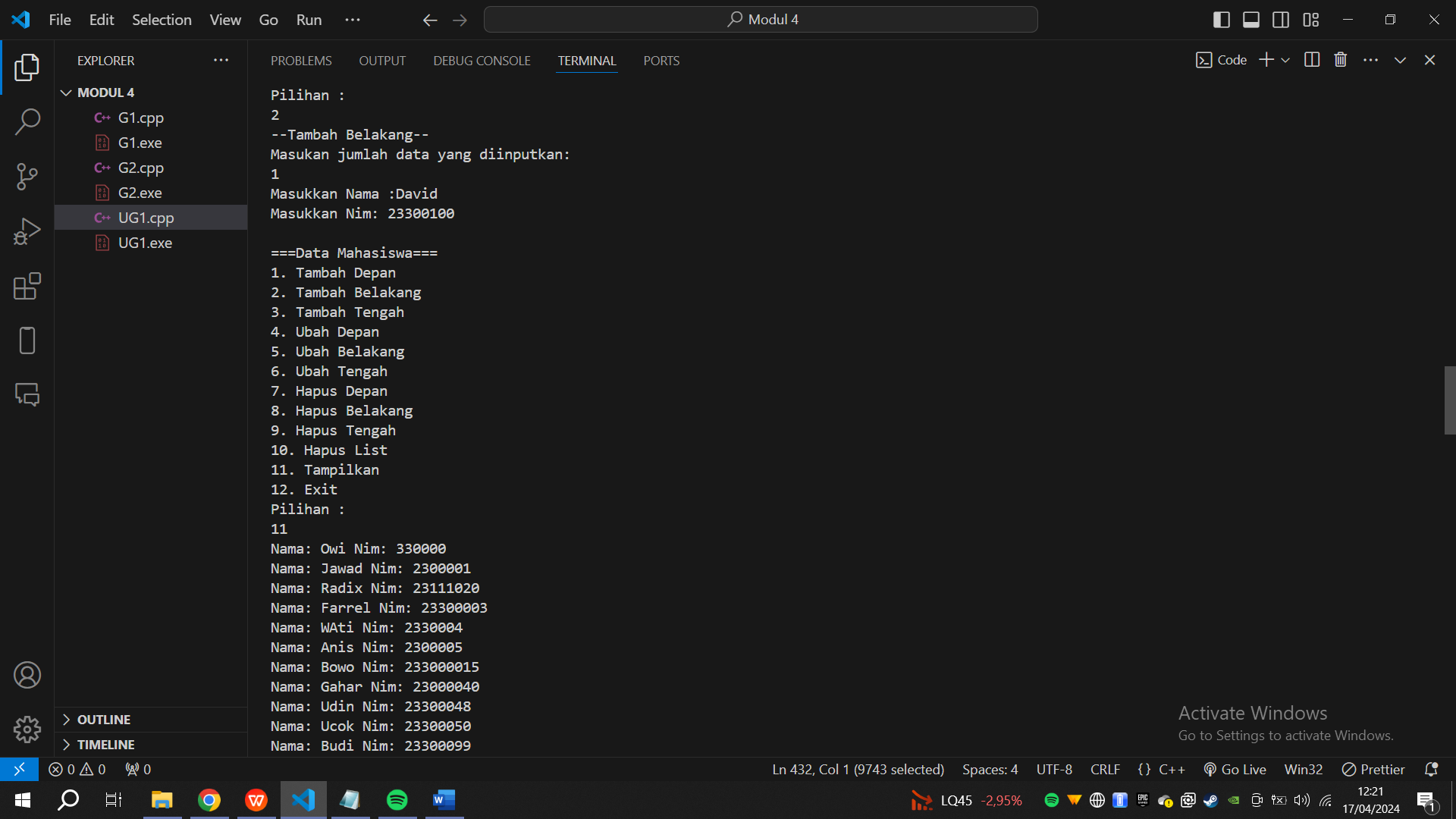
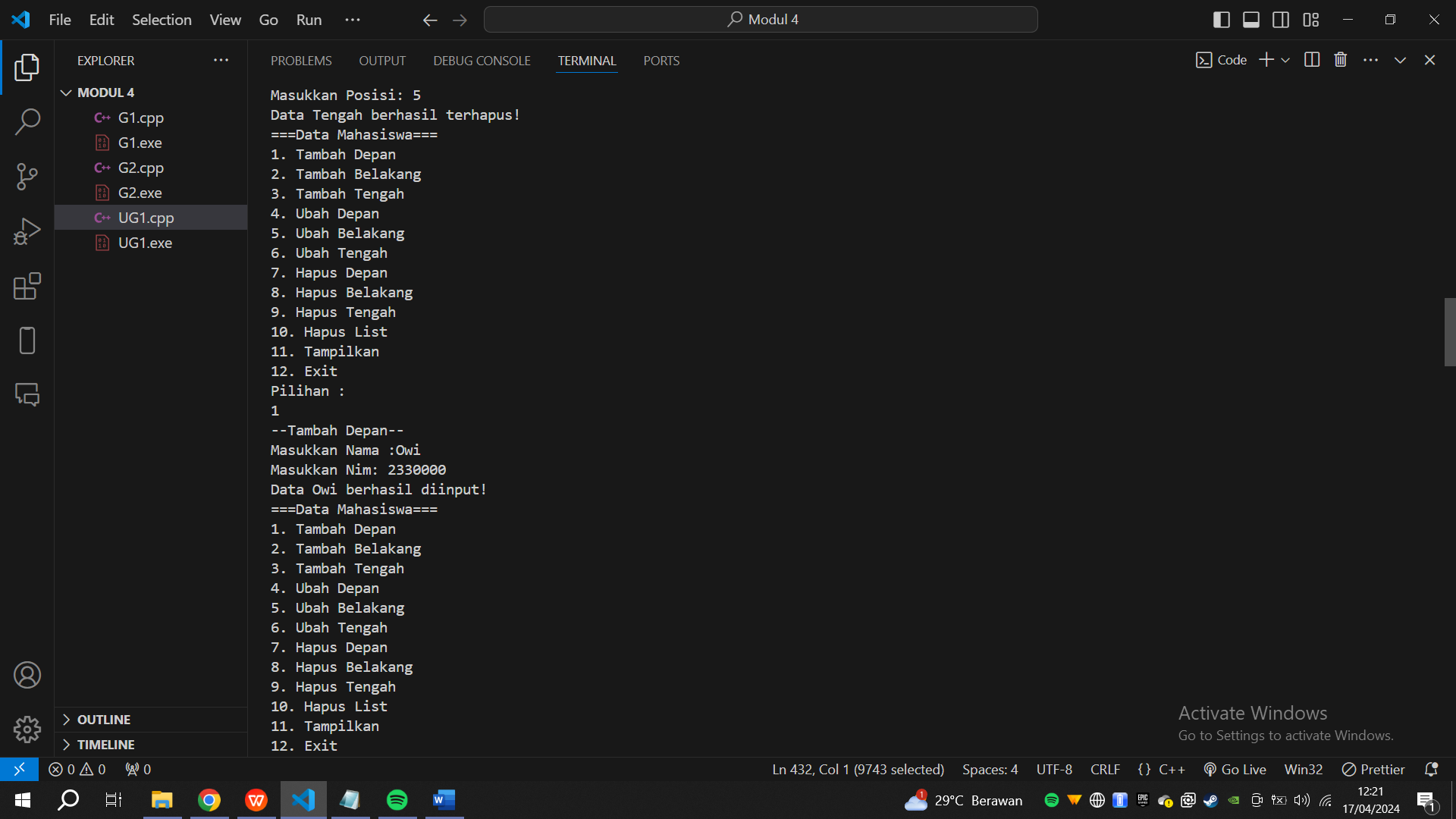
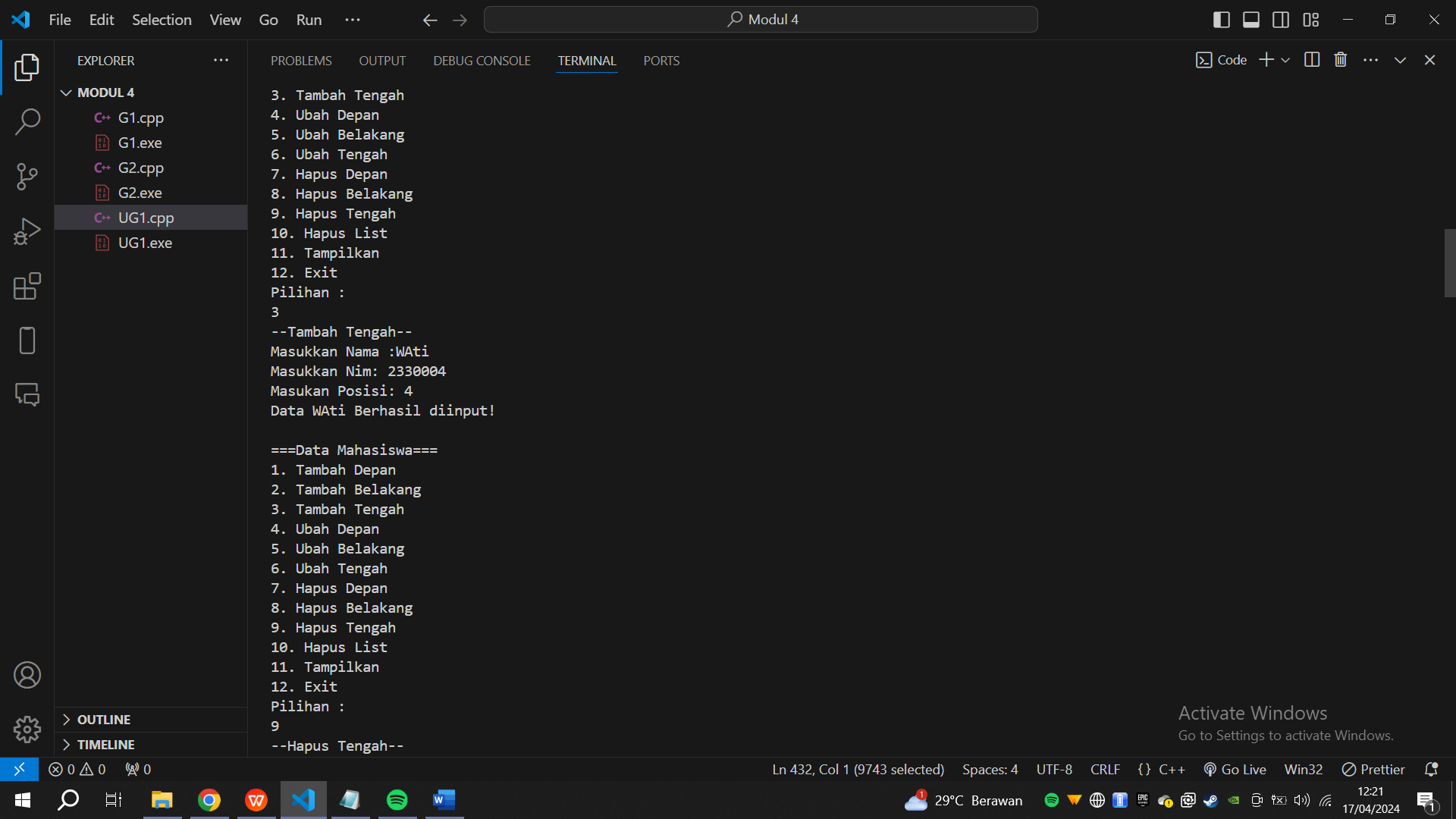
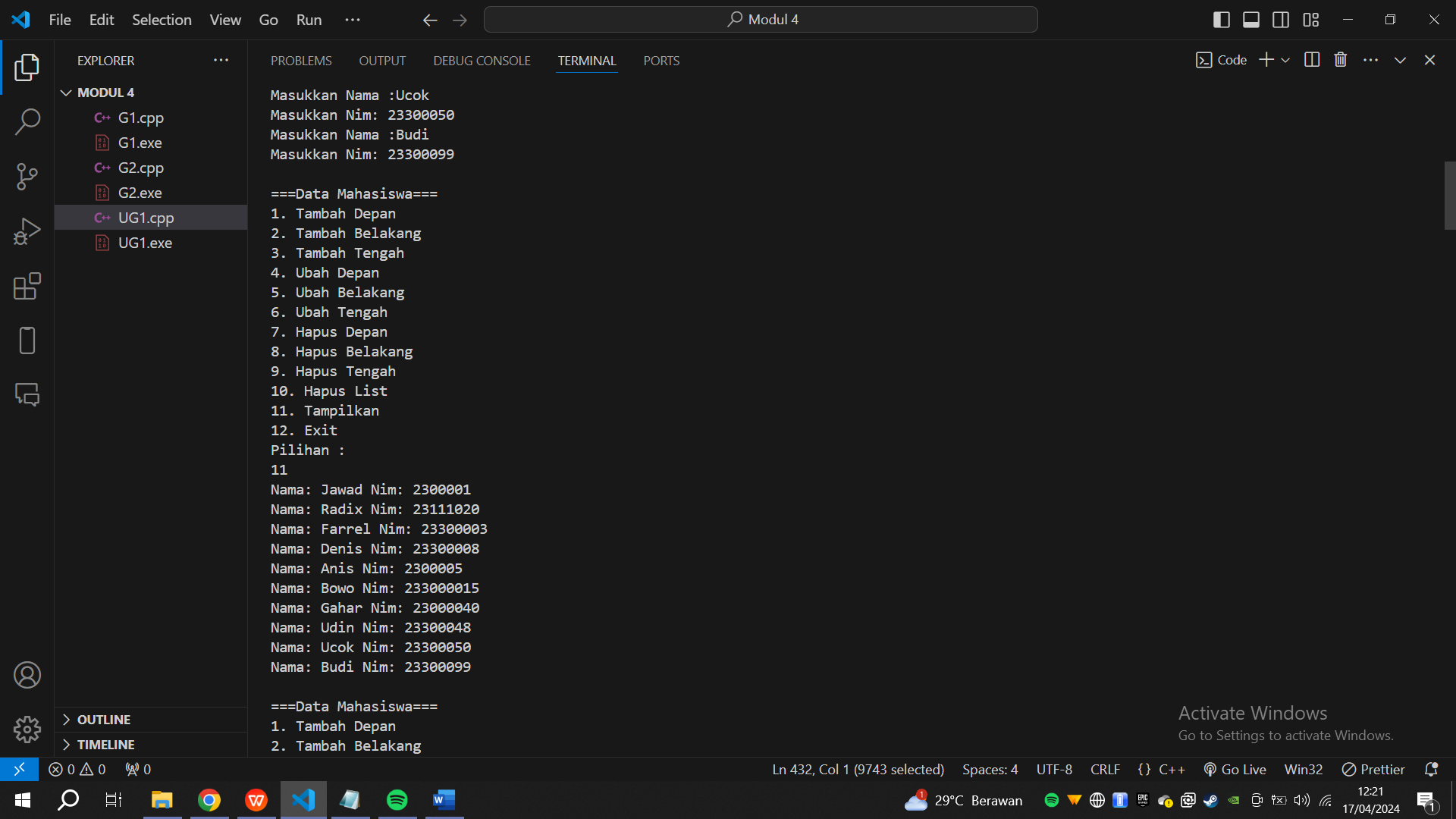
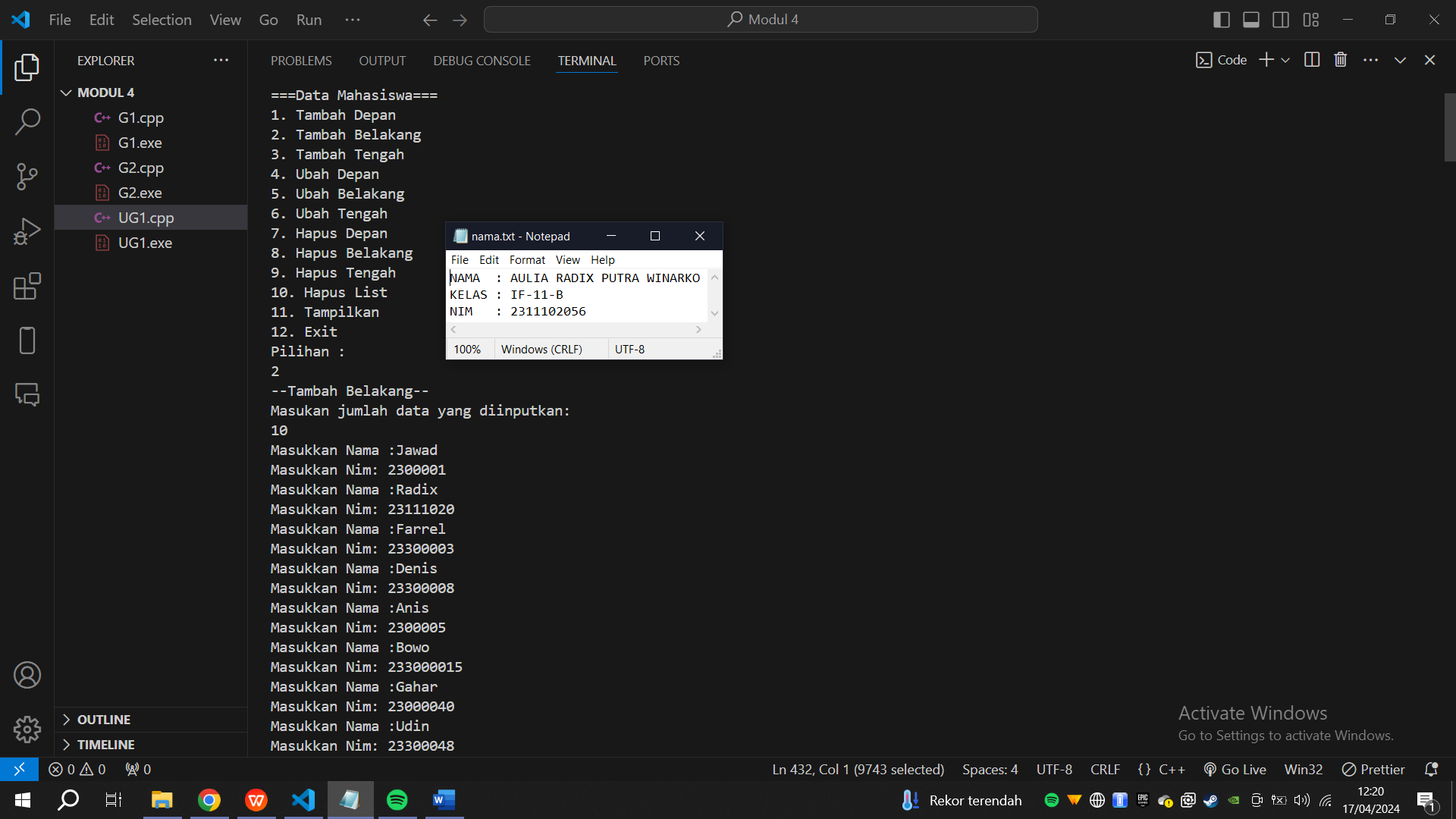
.

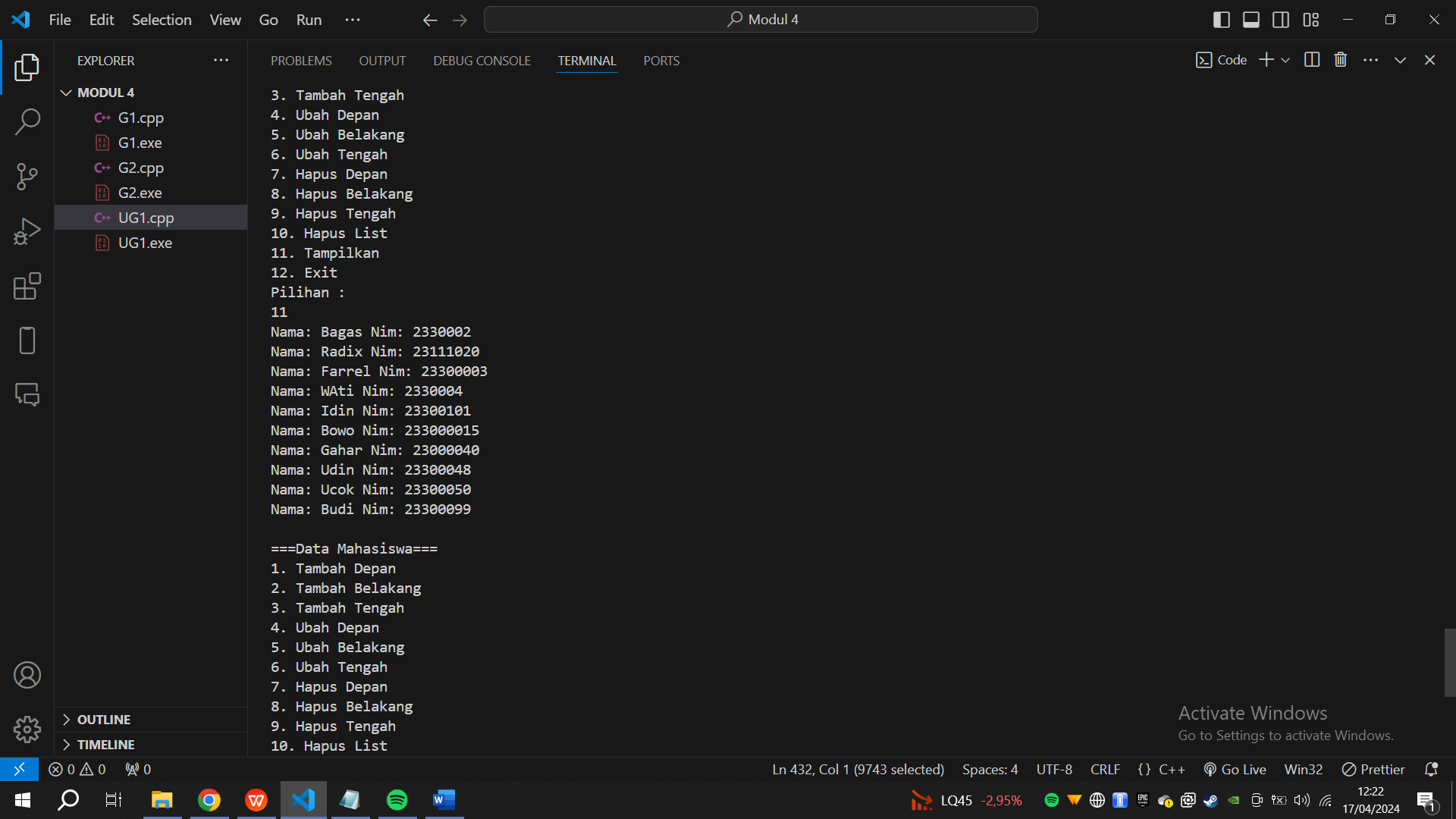
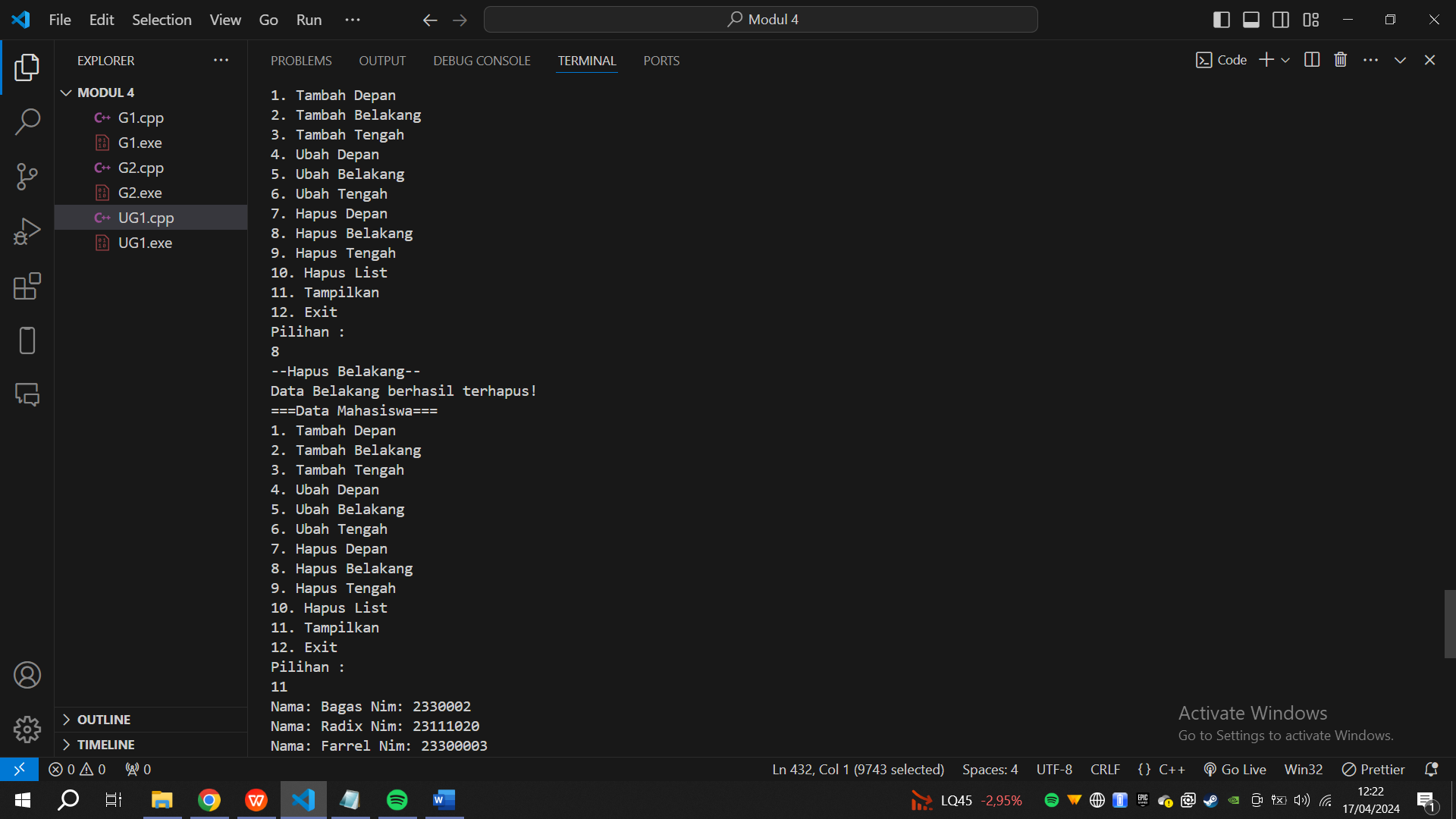
1. **Unguided/Tugas (berisi screenshot source code & output program disertai penjelasannya)**

Unguided 1

|  |
| --- |
| #include <iostream>  using namespace std;  /// PROGRAM SINGLE LINKED LIST NON-CIRCULAR  // Deklarasi Struct Node  struct Node  {      string nama;      int nim;      Node \*next;  };  Node \*head;  Node \*tail;  // Inisialisasi Node  void init()  {      head = NULL;      tail = NULL;  }  // Pengecekan  bool isEmpty()  {      if (head == NULL)          return true;      else          return false;  }  // Tambah Depan  void insertDepan(string nama, int nim)  {      // Buat Node baru      Node \*baru = new Node;      baru->nama = nama;      baru->nim = nim;      baru->next = NULL;      if (isEmpty() == true)      {          head = tail = baru;          tail->next = NULL;      }      else      {          baru->next = head;          head = baru;      }  }  // Tambah Belakang  void insertBelakang(string nama, int nim)  {      // Buat Node baru      Node \*baru = new Node;      baru->nama = nama;      baru->nim = nim;      baru->next = NULL;      if (isEmpty() == true)      {          head = tail = baru;          tail->next = NULL;      }      else      {          tail->next = baru;          tail = baru;      }  }  // Hitung Jumlah List  int hitungList()  {      Node \*hitung;      hitung = head;      int jumlah = 0;      while (hitung != NULL)      {          jumlah++;          hitung = hitung->next;      }      return jumlah;  }  // Tambah Tengah  void insertTengah(string nama, int nim, int posisi)  {      if (posisi < 1 || posisi > hitungList())      {          cout << "Posisi diluar jangkauan" << endl;      }      else if (posisi == 1)      {          cout << "Posisi bukan posisi tengah" << endl;      }      else      {          Node \*baru, \*bantu;          baru = new Node();          baru->nama = nama;          baru->nim = nim;          // tranversing          bantu = head;          int nomor = 1;          while (nomor < posisi - 1)          {              bantu = bantu->next;              nomor++;          }          baru->next = bantu->next;          bantu->next = baru;      }  }  // Hapus Depan  void hapusDepan()  {      Node \*hapus;      if (isEmpty() == false)      {          if (head->next != NULL)          {              hapus = head;              head = head->next;              delete hapus;          }          else          {              head = tail = NULL;          }      }      else      {          cout << "List kosong!" << endl;      }  }  // Hapus Belakang  void hapusBelakang()  {      Node \*hapus;      Node \*bantu;      if (isEmpty() == false)      {          if (head != tail)          {              hapus = tail;              bantu = head;              while (bantu->next != tail)              {                  bantu = bantu->next;              }              tail = bantu;              tail->next = NULL;              delete hapus;          }          else          {              head = tail = NULL;          }      }      else      {          cout << "List kosong!" << endl;      }  }  // Hapus Tengah  void hapusTengah(int posisi)  {      Node \*bantu, \*hapus, \*sebelum;      if (posisi < 1 || posisi > hitungList())      {          cout << "Posisi di luar jangkauan" << endl;      }      else if (posisi == 1)      {          cout << "Posisi bukan posisi tengah" << endl;      }      else      {          int nomor = 1;          bantu = head;          while (nomor <= posisi)          {              if (nomor == posisi - 1)              {                  sebelum = bantu;              }              if (nomor == posisi)              {                  hapus = bantu;              }              bantu = bantu->next;              nomor++;          }          sebelum->next = bantu;          delete hapus;      }  }  // Ubah Depan  void ubahDepan(string nama, int nim)  {      if (isEmpty() == 0)      {          head->nama = nama;          head->nim = nim;      }      else      {          cout << "List masih kosong!" << endl;      }  }  // Ubah Tengah  void ubahTengah(string nama, int nim, int posisi)  {      Node \*bantu;      if (isEmpty() == 0)      {          if (posisi < 1 || posisi > hitungList())          {              cout << "Posisi di luar jangkauan" << endl;          }          else if (posisi == 1)          {              cout << "Posisi bukan posisi tengah" << endl;          }          else          {              bantu = head;              int nomor = 1;              while (nomor < posisi)              {                  bantu = bantu->next;                  nomor++;              }              bantu->nama = nama;              bantu->nim = nim;          }      }      else      {          cout << "List masih kosong!" << endl;      }  }  // Ubah Belakang  void ubahBelakang(string nama, int nim)  {      if (isEmpty() == 0)      {          tail->nama = nama;          tail->nim = nim;      }      else      {          cout << "List masih kosong!" << endl;      }  }  // Hapus List  void clearList()  {      Node \*bantu, \*hapus;      bantu = head;      while (bantu != NULL)      {          hapus = bantu;          bantu = bantu->next;          delete hapus;      }      head = tail = NULL;      cout << "List berhasil terhapus!" << endl;  }  // Tampilkan List  void tampil()  {      Node \*bantu;      bantu = head;      if (isEmpty() == false)      {          while (bantu != NULL)          {              cout << "Nama: " << bantu->nama << " Nim: " << bantu->nim << endl;              bantu = bantu->next;          }      }      else      {          cout << "List masih kosong!" << endl;          return;      }  }  int main()  {      string nama;      int nim;      int pilihan;      int posisi;      do      {          cout << "\n===Data Mahasiswa===" << endl;          cout << "1. Tambah Depan" << endl;          cout << "2. Tambah Belakang" << endl;          cout << "3. Tambah Tengah" << endl;          cout << "4. Ubah Depan" << endl;          cout << "5. Ubah Belakang" << endl;          cout << "6. Ubah Tengah" << endl;          cout << "7. Hapus Depan" << endl;          cout << "8. Hapus Belakang" << endl;          cout << "9. Hapus Tengah" << endl;          cout << "10. Hapus List" << endl;          cout << "11. Tampilkan" << endl;          cout << "12. Exit" << endl;          cout << "Pilihan :" << endl;          cin >> pilihan;          switch (pilihan)          {          case 1:          {              cout << "--Tambah Depan--" << endl;              cout << "Masukkan Nama :";              cin.ignore();              getline(cin, nama);              cout << "Masukkan Nim: ";              cin.ignore();              cin >> nim;              insertDepan(nama, nim);              cout << "Data " << nama << " berhasil diinput!";              break;          }          case 2:          {              cout << "--Tambah Belakang--" << endl;              int jumlah;              cout << "Masukan jumlah data yang diinputkan: ";              cin >> jumlah;              for (int q = 1; q <= jumlah; q++)              {                  cout << "Masukkan Nama :";                  cin.ignore();                  getline(cin, nama);                  cout << "Masukkan Nim: ";                  cin >> nim;                  insertBelakang(nama, nim);              }              break;          }          case 3:          {              cout << "--Tambah Tengah--" << endl;              cout << "Masukkan Nama :";              cin.ignore();              getline(cin, nama);              cout << "Masukkan Nim: ";              cin >> nim;              cout << "Masukan Posisi: ";              cin >> posisi;              insertTengah(nama, nim, posisi);              cout << "Data " << nama << " Berhasil diinput!" << endl;              break;          }          case 4:          {              cout << "--Ubah Depan--" << endl;              cout << "Masukkan Nama :";              cin.ignore();              getline(cin, nama);              cout << "Masukkan Nim: ";              cin >> nim;              ubahDepan(nama, nim);              break;          }          case 5:          {              cout << "--Ubah Belakang--" << endl;              cout << "Masukkan Nama :";              cin.ignore();              getline(cin, nama);              cout << "Masukkan Nim: ";              cin >> nim;              ubahBelakang(nama, nim);              break;          }          case 6:          {              cout << "--Ubah Tengah--" << endl;              cout << "Masukkan Nama :";              cin.ignore();              getline(cin, nama);              cout << "Masukkan Nim: ";              cin >> nim;              cout << "Masukkan Posisi: ";              cin >> posisi;              ubahTengah(nama, nim, posisi);              break;          }          case 7:          {              cout << "--Hapus Depan--" << endl;              hapusDepan();              cout << "Data Depan berhasil terhapus!";              break;          }          case 8:          {              cout << "--Hapus Belakang--" << endl;              hapusBelakang();              cout << "Data Belakang berhasil terhapus!";              break;          }          case 9:          {              cout << "--Hapus Tengah--" << endl;              cout << "Masukkan Posisi: ";              cin >> posisi;              hapusTengah(posisi);              cout << "Data Tengah berhasil terhapus!";              break;          }          case 10:          {              clearList();              break;          }          case 11:          {              tampil();              break;          }          case 12:          {              cout << "Terima Kasih!" << endl;          }          default:          {              cout << "Pilihan tidak Valid!" << endl;              break;          }          }      } while (pilihan != 12);      return 0;  } |

Screenshots Output





Deskripsi:

Kode C++ di atas adalah implementasi dari sebuah program yang menggunakan linked list untuk menyimpan dan mengelola data Mahasiswa. Program ini memberikan fungsionalitas untuk mengelola data Mahasiswa seperti menambah, mengubah, dan menghapus data, serta menampilkan data Mahasiswa yang sudah tersimpan dalam linked list.

1. Referensi[Bob

Karumanchi, N. (2016). *Data Structures and algorithms made easy: Concepts, problems, Interview Questions*. CareerMonk Publications.sebesar 15%