Tugas Pendahuluan Modul 4

STRUKTUR DATA - Ganjil 2024/2025

"Single\_Linked\_List\_Bagian\_1"

Ketentuan Tugas Pendahuluan

* 1. Tugas Pendahuluan dikerjakan secara Individu.
  2. TP ini bersifat WAJIB, tidak mengerjakan = PENGURANGAN POIN JURNAL / TES ASESMEN.
  3. Hanya MENGUMPULKAN tetapi TIDAK MENGERJAKAN = PENGURANGAN POIN JURNAL / TES ASESMEN.
  4. Deadline pengumpulan TP Modul 2 adalah Senin, 30 September 2024 pukul 07.30 WIB.
  5. TIDAK ADA TOLERANSI KETERLAMBATAN, TERLAMBAT ATAU

TIDAK MENGUMPULKAN TP MAKA DIANGGAP TIDAK MENGERJAKAN.

* 1. DILARANG PLAGIAT (PLAGIAT = E).
  2. Kerjakan TP dengan jelas agar dapat dimengerti.
  3. Codingan diupload di Github dan upload Laporan di Lab menggunakan format PDF dengan ketentuan:

TP\_MOD\_[XX]\_NIM\_NAMA.pdf

CP (WA):

* + Andini (082243700965)
  + Imelda (082135374187)

**SELAMAT MENGERJAKAN^^**

# LAPORAN PRAKTIKUM

# PERTEMUAN 4

**Single\_Linked\_List\_Bagian\_1**



## Nama :

Zulfa Mustafa Akhyar Iswahyudi (2311104010)

## Dosen :

Yudha Islami Sulistya

# PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

# FAKULTAS INFORMATIKA

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

1. Tujuan

Untuk melatih kompetensi Mahasiswa untuk memperdalam skill pemrograman C++

1. Tools

Codeblocks, VSCode, Github

**TUGAS PENDAHULUAN**

1. **List.h**

Secara keseluruhan, dibawah ini adalah program untuk melakukan *reversal (pembalikan)* urutan beberapa inputan yang dilakukan oleh user ke sintaks program. Kita bagi programnya menjadi tiga bagian. Satu untuk sub-class **list.h**, satu untuk menampung seluruh method dan fungsi dalam **list.cpp**, dan terakhir adalah **main.cpp** yang menjadi class operasional utama.

=================================================

Untuk sub-class yaitu **list.h** adalah sub-class yang mendeklarasikan **definite/fungsi** dari elemen *infotype* dan *elmlist* yang akan menjadi alamat data dari **linkedlist**. Jangan lupa untuk konfigurasi pertama kali yang harus dilakukan adalah mendefinisikan fungsi **first(L)** yang dimana sebagai settingan default suatu nilai dalam barisan data.

Jadi untuk inisiasi pertama kali dalam **linkedlist**, data yang akan terbaca duluan adalah data paling depan dalam **linkedlist** data, jadi begitu simpelnya. Selanjutnya ada fungsi **next(p)** dan **info(p)**.

**P** adalah variabel data tertentu,

Sementara fungsi **next** adalah fungsi untuk memajukan variabel tertentu dalam **linkedlist** sebanyak 1 langkah. Sedangkan

**info** adalah fungsi untuk menampilkan hasil kalkulasi program terhadap data **linkedlist**. Lalu kita mulai membuat **constructor** pada *elmlist* yang diisikan fungsi *infotype* dengan inisialisasi **info**, lalu fungsi *address* dengan inisialisasi **next** dan *List* yang diisikan fungsi address dengan inisialisasi first.

=================================================

Langkah Paling akhir disini adalah membuat method untuk menciptakan **linkedlist**, membuat fungsi pengalokasian data, method untuk memasukkan data pada **linkedlist** berdasarkan alamatnya, dan terakhir adalah method untuk menampilkan data **linkedlist**-nya.

Glosarium :

* List &L : Ini adalah sintaks untuk pemanggilan **linkedlist** yang dideklarasi beserta

dengan alamat **linkedlist** data-nya.

* Infotype x : Ini adalah sintaks untuk menargetkan serta mengidentifikasi

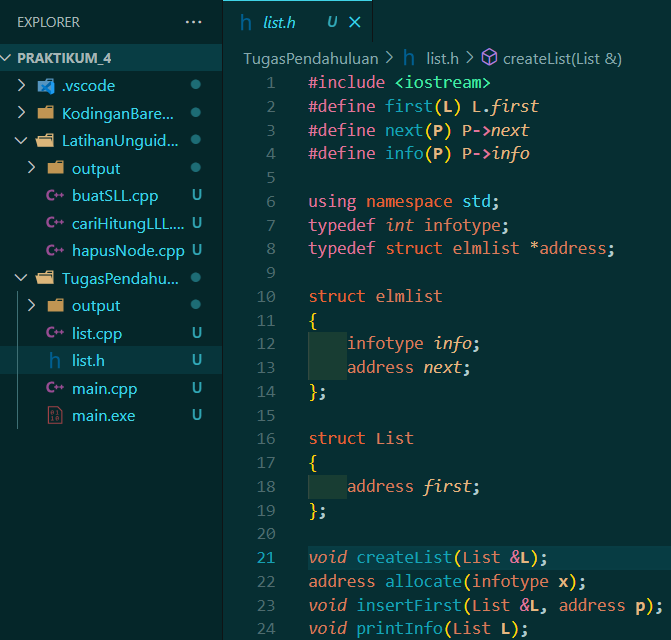
nilai/data tertentu pada sebuah **linkedlist** yang bisa dimanipulasi

seperti dihapus, dimajukan 1,2,3 langkah, atau dimundurkan

sebanyak langkah sesuai inputan seperti halnya memajukan data.

* Address p : Ini adalah sintaks untuk pencarian alamat dari suatu data(p)

tertentu agar bisa dialokasikan ke posisi tertentu dalam **linkedlist**.



1. **List.cpp**

Lanjut ke file untuk implementasi variabel definisi beserta variabel construct yang sudah dibuat, yaitu list.cpp. Kita perlu memanggil sub-class list.h agar semua pendeklarasian dapat terdeteksi dan sistem dapat mengidentifikasi dengan baik keseluruhan deklarasinya.

=================================================

Buat method **createList** untuk menciptakan **linkedlist** yang parameter-nya itu **<List &L>** yang memuat fungsi insialisasi sebuah List **Linkedlist**. Didalam method-nya tambahkan juga definite/fungsi definisi **<first(L)>** yang kita atur kosong diawal.

=================================================

Selanjutnya adalah method khusus untuk pengalokasian sebuah nilai berdasarkan fungsi definisi alamat, parameter-nya **<infotype x>** yang memuat fungsi untuk menargetkan sebuah variabel tertentu.

Didalam method-nya berisi deklarasi baru dari constructor elmlist yang diinisialisasi fungsi **<address p>**. Kemudian atur default data(p) agar bisa ditampilkan dan data selanjutnya adalah kosong dulu. Lalu buat pengembalian nilai yang akan menjadi output.

=================================================

Selanjutnya buat method **insertFirst** agar data inputan bisa masuk kedalam **linkedlist** yang sudah dibuat sebelumnya. Dengan dua parameter <List &L> dan <address p> kita sudah dapat identifikasi data beserta **linkedlist**-nya yang akan kita manipulasi. Didalam method ini kita hanya perlu mengatur bahwa data(p) akan ditaruh diurutan paling awal dideapan **linkedlist**.

=================================================

Terakhir adalah method **printInfo** untuk menampilkan keseluruhan **linkedlist**, parameter utamanya adalah **<List L>**. Karena kita ingin menampilkan keseluruhan data **linkedlist**, maka kita gunakan notasi **L** dan bukan **&L**. Didalamnya adalah pengkondisian jika data **linkedlist** tidak kosong maka **linkedlist** akan ditampilkan dalam output. Satu per satu data akan ditampilkan seluruhnya dengan fungsi **<next(p)>** untuk membacca data selanjutnya setelah data pertama.

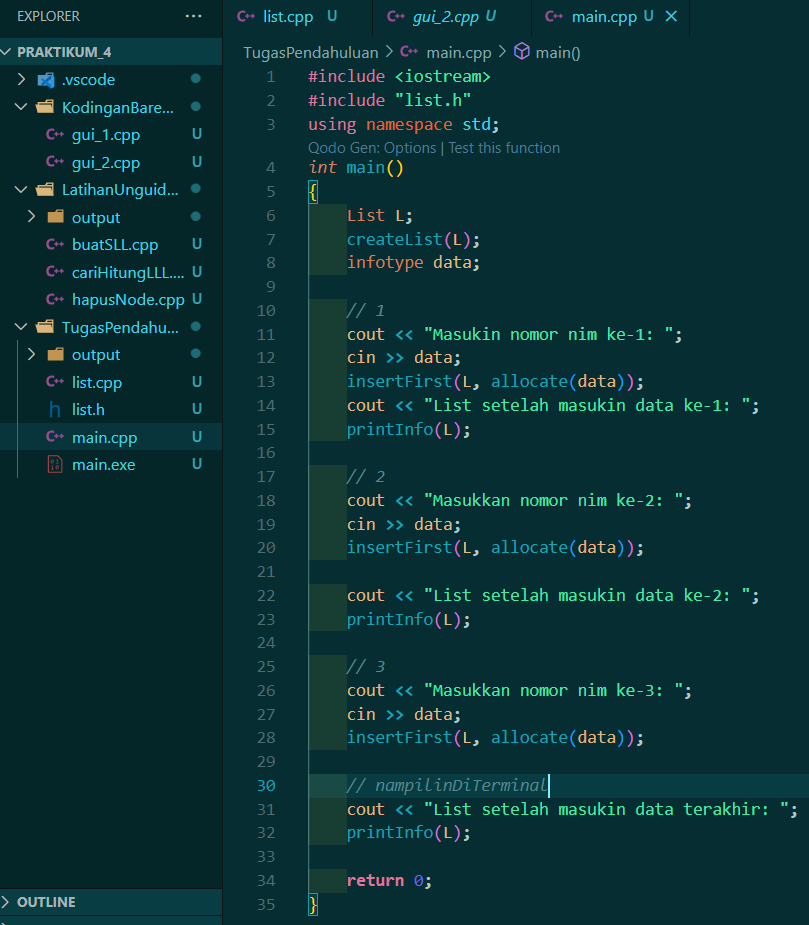


1. **Main.cpp**

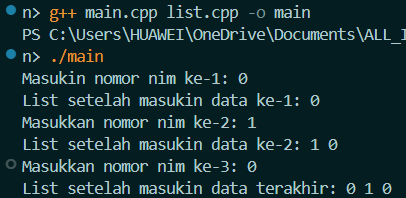
Masuk Class utama yang mengoperasikan seluruh method dan fungsi, main.cpp. Kita perlu meng-import sub-class **list.h** agar seluruh deklarasinya dapat terdeteksi dan diimplementasikan dengan lancar. Deklarasikan **<List L>** lalu buat **linkedlist** dengan fungsi **createList** dengan inisialisasi **L**, kemudian deklarasikan variabel **infotype** untuk menargetkan data didalam **linkedlist**.

=================================================

Saatnya mengeksekusi seluruh deklarasi, **sekarang kita buat fugnsi inputan dari user dengan inisiasi data inputannya akan dialokasikan kedalam linkedlist dengan fungsi insertFirst, kemudian ditampilkan di Output, ulangi proses ini sampai tiga kali**. Setelahnya buat return untuk mengembalikan nilai semuanya, alias membuat output terminal.



Output :



**Latihan – UNGUIDED**

1. **Membuat Single Linked List**

Kali ini adalah program untuk menciptakan sebuah barisan **linkedlist** berbasis SLL. Sebelum kita menginputkan secara manual data-data yang ingin kita bariskan, lebih baik buat dulu method-method penting untuk menambahkan data didepan, menambahkan data dibelakang, juga jangan lupa buat method untuk menampilkan barisan list **Linkedlist**-nya.

Untuk deklarasi awal ada constructor untuk Node yang isi konfigurasi-nya ada variabel data yang di-set dengan integer dan Node itu sendiri yang di-set \*next agar data-data inputan bisa masuk kedalam barisan list **Linkedlist**.

Menurut saya pribadi, Node itu adalah variabel untuk membentuk struktur **Linkedlist**. Jadi barisan list data yang kita buat sumber utamanya adalah Node yang dideklarasikan diawal.

==================================================

Untuk method ‘depan’, kita perlu **masukkan Node beserta alamatnya dan variabel baru yaitu nilai dengan set integer kedalam parameter-nya agar data dapat diidentifikasi dengan inputan berupa angka**. Didalam method ini kita perlu deklarasikan Node menjadi variabel fungsi baru supaya struktur **linkedlist** tetap solid saat ada tambahan data baru.

Atur NewNode yang sudah dideklarasikan ulang tadi agar data diidentifikasi sebagai nilai dan fungsi definisi next sebagai head. Head ini selain dia adalah data pertama dalam linkedlist, diidentifikasi sebagai newNode juga. **Artinya tujuan dari korelasi antara fungsi definisi next dengan variabel Head(nilai pertama dalam linkedlist) adalah agar linkedlist dapat terus meregenerasi indeks-nya saat inputan baru masuk kedalam linkedlist**.

==================================================

Selanjutnya ada method ‘belakang’. Saya ga perlu panjang lagi jelasinnya, tapi intinya sintaks dan deklarasi didalam method ini jelas sama seperti method ‘depan’ hanya saja perbedaannya adalah deklarasi newNode dengan next adalah di-set menjadi nullptr yang dimana itu merujuk ke nilai paling akhir/belakang dalam **linkedlist**. Artinya data yang diinputkan tidak akan berada didepan data sebelumnya.

Dari kekosongan posisi data ini, kita perlu kondisikan kalo nilai pertama dalam **linkedlist** ada di paling belakang **linkedlist**, maka ‘head’ ini akan tetap meregenerasi indeks **linkedlist**. Namun jika tidak kosong, maka variabel ‘head’ akan diinisialisasikan sebagai Node sementara. Kondisikan lagi selama data selanjutnya setelah Node sementara itu bukan dibelakang **linkedlist**, maka Node sementara tadi akan menjadi indeks baru dalam **linkedlist**, artinya Node sementara tadi berubah menjadi nilai asli didalam **linkedlist**. Namun posisinya akan berada dibelakang nilai awal.

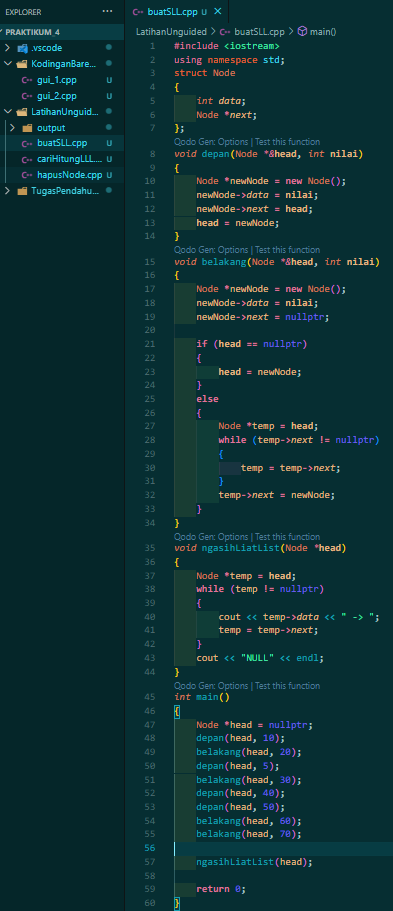
==================================================

Lalu ada method ‘ngasihLiatList’. Gampangnya ini method untuk menampilkan seluruh baris **linkedlist**. Cukup parameter-kan Node dengan nilai pertamanya, kemudian didalam method ini kondisikan selama Node sementara bukan nilai paling belakang, maka data **linkedlist** akan ditampilkan secara urut.

==================================================

Terakhir adalah program untuk inputannya. **Kita ga pake cout untuk outputnya, tetapi cukup panggil method yang sudah dibuat lalu deklarasikan jumlah nilainya dalam bentuk angka**. Jangan lupa deklarasikan ‘head’ yang mewakili variabel nilai yang akan masuk ke dalam **linkedlist**.

Sama diatasnya itu deklarasiin dulu Node-nya dengan alamat nilainya. Diakhir line, panggil method ‘ngasihLiatList dengan parameter data didalamnya agar terminal menampilkan barisan **linkedlist**-nya’



Output :



1. **Menghapus Node pada Linked List**

Selanjutnya adalah sintaks program untuk menghapus sebuah Node atau data dalam LinkedList. Class kali ini tidak berbeda dengan Class sebelumnya, namun ada penambahan method baru, yaitu method **hapusNode**.

==================================================

Parameter-kan dulu deklarasi Node beserta seluruh alamat data **linkedlist**, serta deklarasi nilai yang di-set integer. **Didalamnya ada pengkondisian bercabang yang rumit, pertama adalah pengkondisian dimana data pertama di linkedlist ada di paling belakang, maka method akan mengembalikan nilai**.

==================================================

Pada pengkondisian berikutnya**, jika data tertentu dalam barisan linkedlist ditemukan nilai-nya, maka data tersebut akan menjadi Node sementara kemudian dihapus dari linkedlist-nya**, lalu method akan mengembalikan nilai.

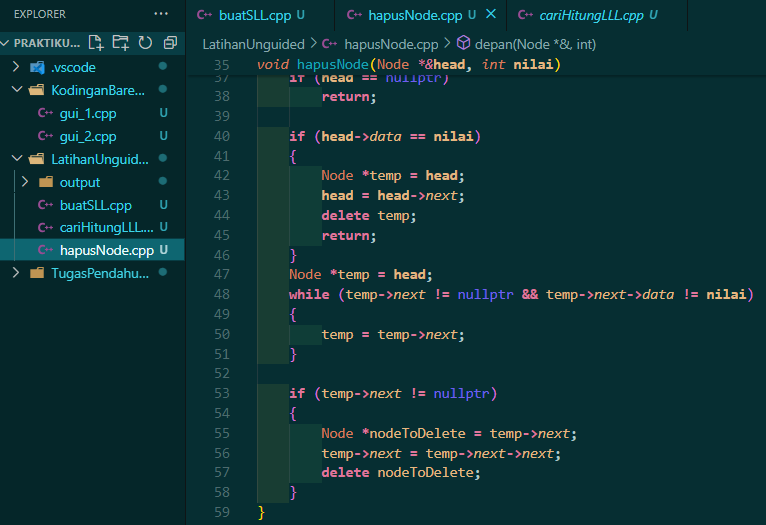
==================================================

Sekarang deklarasikan bahwa Node sementara adalah data dalam list **linkedlist** juga. **Buat perulangan dimana selama nilai sementara dari Node tadi tidak berada dibelakang linkedlist dan nilai sementara dalam keseluruhan data dalam linkedlist tidak ditemukan nilai yang ditargetkan**, maka nilai sementara tadi akan dimajukan posisinya untuk menggantikan posisi data yang telah hilang.

==================================================

Kondisikan lagi jika nilai sementara dan nilai seterusnya tidak ada dibelakang **linkedlist**, maka Node akan mendeklarasikan nilai sementara ini dengan **nodeToDelete** yang nantinya akan menghapus nilai yang sesuai.

Lalu pada penginputan manual di **main()**, kita bisa tambahkan sintaks baru yaitu ‘hapusNode’, kemudian dekalrasikan variabel head dengan data yang ingin dihapus. Disitu kita contohkan 50, maka di terminal angka 50 sudah hilang.



Output :



1. **Mencari dan Menghitung Panjang Linked List**

Untuk terakhir kalinya dalam laporan ini, adalah Class untuk mencari sebuah nilai tertentu beserta panjang list **linkedlist**-nya.

==================================================

Pada method **‘cariNode‘,** kita akan parameter-kan Node dengan seluruh data dan deklarasi nilai bertipe integer. Kondisikan langsung apabila nilai dalam data **linkedlist** sesuai dengan nilai yang kita cari, maka method akan mengembalikan nilai **True yang artinya nilai yang ditargetkan telah ditemukan.**

Namun jika pendeteksian dilakukan terus pada seluruh nilai seterusnya dan sampai akhir tidak ada, maka akan **False yang artinya nilai tak ditemukan.**

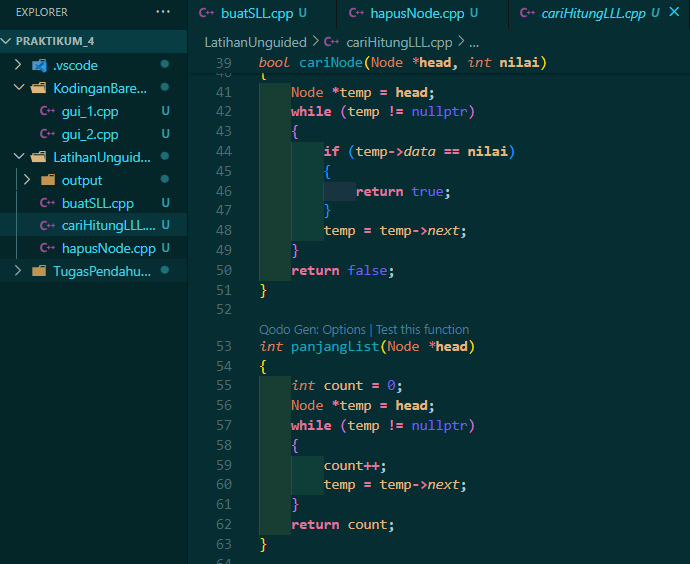
==================================================

Selanjutnya adalah method **‘panjangList’**, cukup Node beserta seluruh datanya sebagai parameter. Lalu set awal ‘count’ jadi nol sebagai awal mula pengecekan, kemudian atur nilai Node awal dimulai dari nilai pertama dalam **linkedlist**.

Kondisikan selama nilai yang dicari tak ada dibelakang **linkedlist**, maka fungsi ‘count’ tadi akan terus bertambah untuk mencari nilai yang kita cari. Setelah ditemukan maka method akan mengembalikan nilai ‘count’.

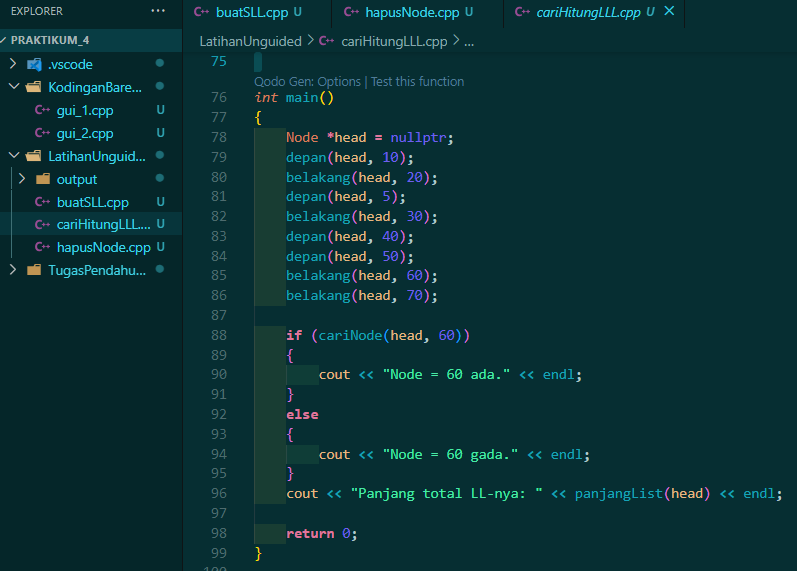
==================================================

**Disini ‘count’ mewakili pencarian nilai yang kita cari sebelumnya.**



Ah iya, jangan lupa juga pada **main()**, kita buat inputan manualnya dengan nilai angka-angka yang masih sama. **Namun kondisikan apabila method yang mencari nilai 60 menemukan nilainya, maka akan terbentuk output identifikasi yang sesuai nilainya**. Namun jika tidak, maka program akan mengatakan kalo Node = 60 tidak ditemukan.

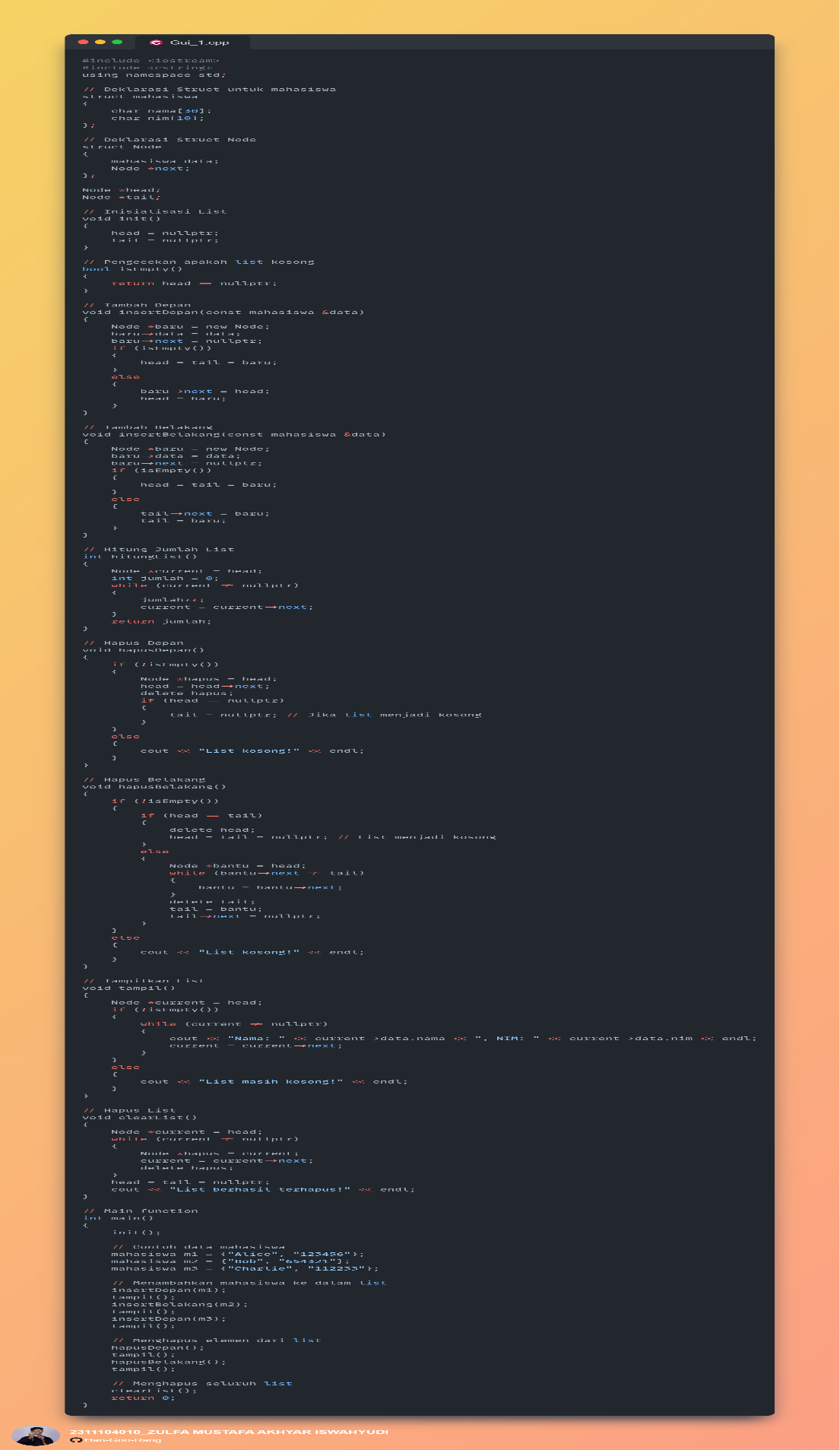
Terakhir, buat output yang menampilkan keseluruhan panjang **linkedlist**-nya.



Output :



**SOURCE CODE WAKTU DI KELAS – KODINGAN ASPRAK**





**Semoga Selalu diberi kemudahan^^**