Program Studi Teknik Elektro ITB

Nama Kuliah (Kode) : Praktikum Pemecahan Masalah dengan C (EL2208)
Tahun / Semester : 2023-2024 / Genap
Modul : 6 – Linked List
Nama Asisten / NIM : Agape D'sky / 23223031
Nama Praktikan / NIM : Pradigta Hisyam Ramadhan / 18322008

BCL / Log Praktikum

Alasan Pemilihan Soal

Pada praktikum ini, penulis memilih soal pertama karena program memiliki input integer dan bukan string. Berdasarkan pengalaman penulis pada praktikum modul-modul sebelumnya, soal dengan input string biasanya akan lebih sulit untuk dikerjakan. Selain itu, fungsi-fungsi yang harus digunakan pada program juga sudah tersedia di internet, sehingga penulis hanya perlu memodifikasi *source code* sesuai dengan spesifikasi soal.

Strategi Awal/Rancangan Algoritma

1. Pengambilan input integer: Pada awalnya, program akan menerima input dari pengguna berupa integer. Input yang dimaksud adalah jumlah linked list, jumlah elemen pada linked list, dan isi dari elemen tersebut. Input dianggap selalu valid pada persoalan ini.

2. Menggabungkan elemen linked list: Setelah program menerima input elemen-elemen dari linked list, maka selanjutnya program akan menggabungkan semua elemen menjadi satu linked list utama. Struktur dari linked list yang digunakkan adalah *int data* dan *reference* dari node selanjutnya.

3. Mengurutkan elemen linked: Setelah mendapatkan linked list utama, maka program akan mengurutkan data-data pada linked list tersebut dari yang paling kecil hingga yang paling besar.

4. Memisahkan bilangan genap dan ganjil: Setelah elemen linked list terurut, maka program akan memisahkan bilangan genap dan ganjil. Bilang ganjil akan berada di sebelah kiri dan bilangan genap akan berada di sebelah kanan.

Kejadian Saat Praktikum

Selama praktikum, penulis menghabiskan waktu sekitar 15 menit hanya untuk memilih soal. Setelah memilih soal pertama yang akan dikerjakan, penulis mengalami kesulitan untuk menerima input dari pengguna. Awalnya, penulis mengira bahwa input yang diberikan pengguna berupa *array of integer* yang nantinya akan dimasukkan ke dalam linked list. Akan tetapi, setelah bertanya dengan asisten praktikum, penulis menyadari bahwa maksud dari soal adalah program menerima input yang kemudian input tersebut langsung dimasukkan ke dalam linked-list. Selain itu, penulis juga mengalami kesulitan dalam menuliskan sintaks penerimaan input apabila input integer yang diberikan memiliki spasi. Permasalahan dapat selesai setelah penulis bertanya kepada asisten dan mendapatkan jawaban bahwa spasi juga dianggap sebagai enter pada fungsi *scanf()*.

Setelah menambahkan data-data pada node, penulis tinggal mengurutkan dan memisahkan bilangan ganjil-genap pada linked list. Untuk melakukan implementasi tersebut, penulis membuat fungsi yang algoritmanya sudah ada di internet. Penulis hanya memodifikasi sedikit kode tersebut sehingga memenuhi spesifikasi persoalan.

Tahapan Strategi (*milestone*)

1. Penambahan elemen linked list: Jadi, pada fungsi *add()* program, input yang diberikan akan langsung dimasukkan ke dalam linked list tanpa harus menjadikannya *array of integer*. Peran dari *jumlah*

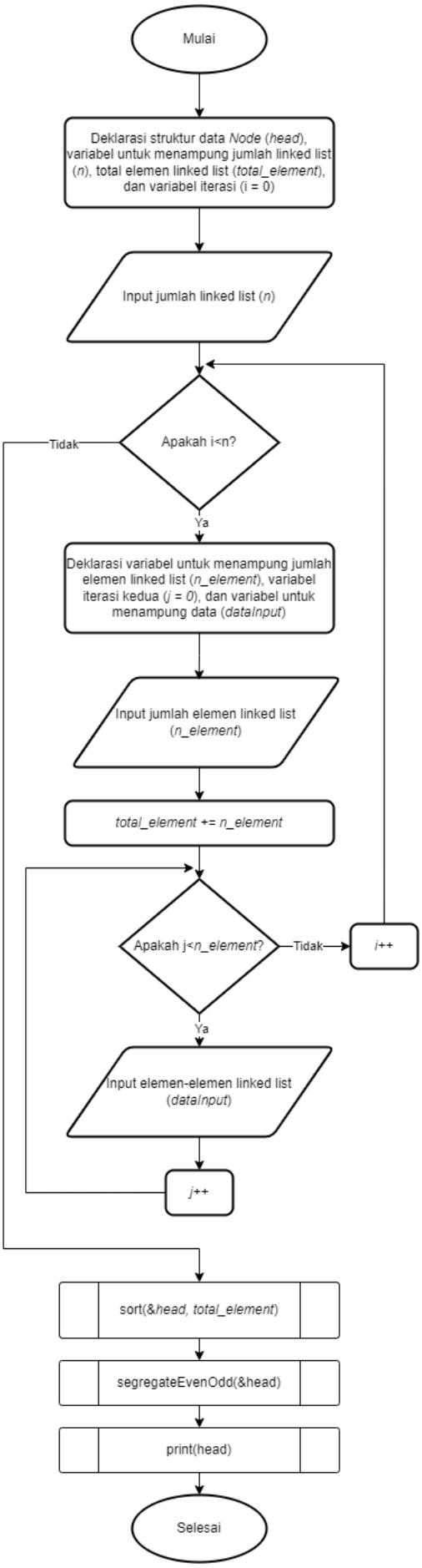
linked list dan *jumlah elemen linked list* pada tampilan program hanya menentukan jumlah data total yang ada pada linked list.

2. Proses mengurutkan dan memisahkan linked list: Linked list yang diperoleh penulis sudah merupakan gabungan dari dua linked list, sehingga penulis tidak memerlukan fungsi untuk menggabungkan dua buah linked list. Selanjutnya, program akan mengurutkan terlebih dahulu bilangan-bilangan yang menjadi data pada linked list. Setelah terurut, maka program akan memisahkan deret bilangan ganjil di sebelah kiri dan deret bilangan genap di sebelah kanan.

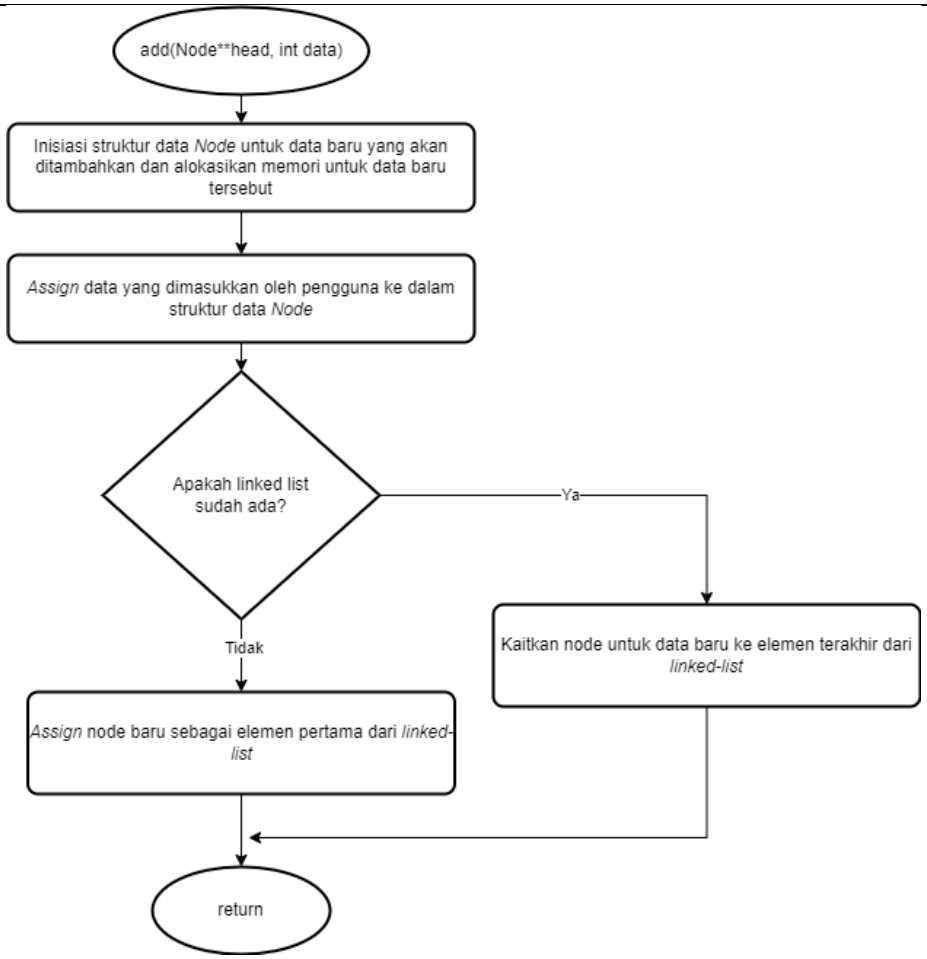
Proses Debugging

Pada saat *source code* program di push di repository github, penulis mengalami dua kesalahan pada test case keempat dan ketujuh. Test case tersebut memiliki input elemen linked list yang tidak memiliki salah satu deret bilangan, baik genap maupun ganjil. Untuk memecahkan masalah tersebut, penulis melakukan proses *debugging* pada fungsi *segregateEvenOdd()* sehingga dapat mengecek apakah ada elemen ganjil/genap pada linked list. Jika tidak ada, maka program akan langsung menampilkan data-data dari linked list. Namun terdapat kejadian menarik saat melakukan proses *debugging* fungsi *segregateEvenOdd()*, yaitu penulis masih meng-*command* fungsi tersebut pada program *main* sehingga penulis tidak dapat segera mendapatkan hasil yang benar sampai hampir 5 menit waktu praktikum tersisa.

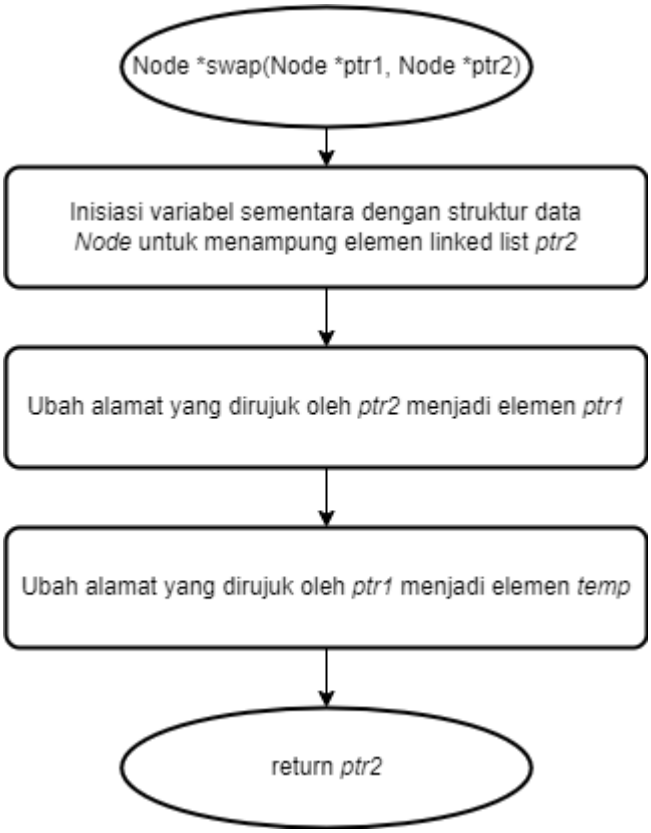
Diagram Alir/*flowchart*



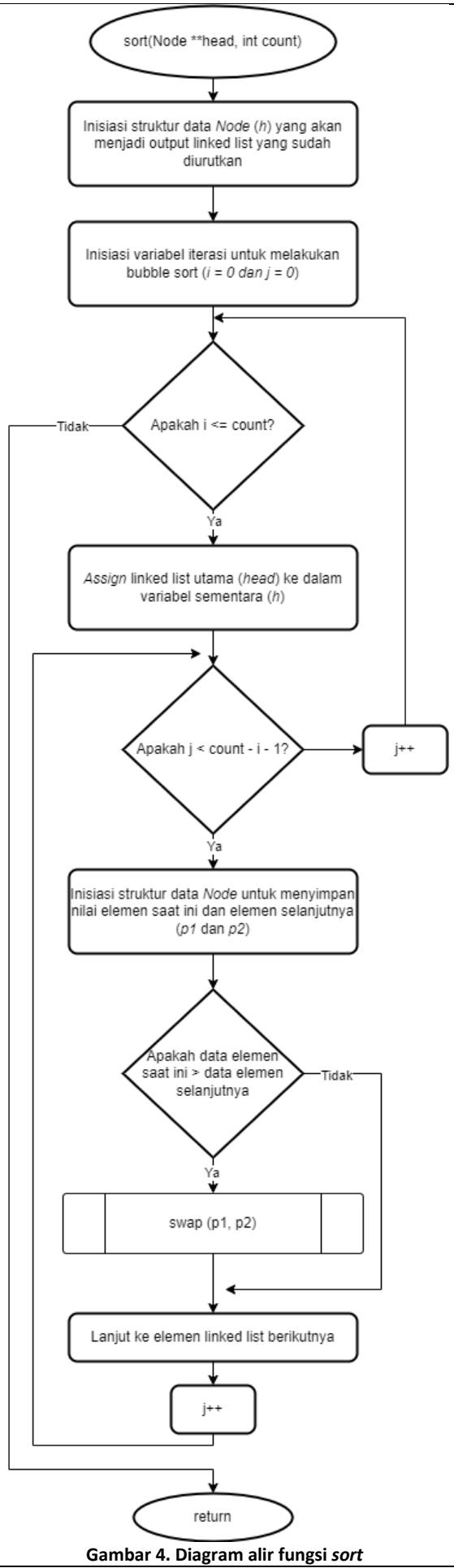
Gambar 1. Diagram alir fungsi utama



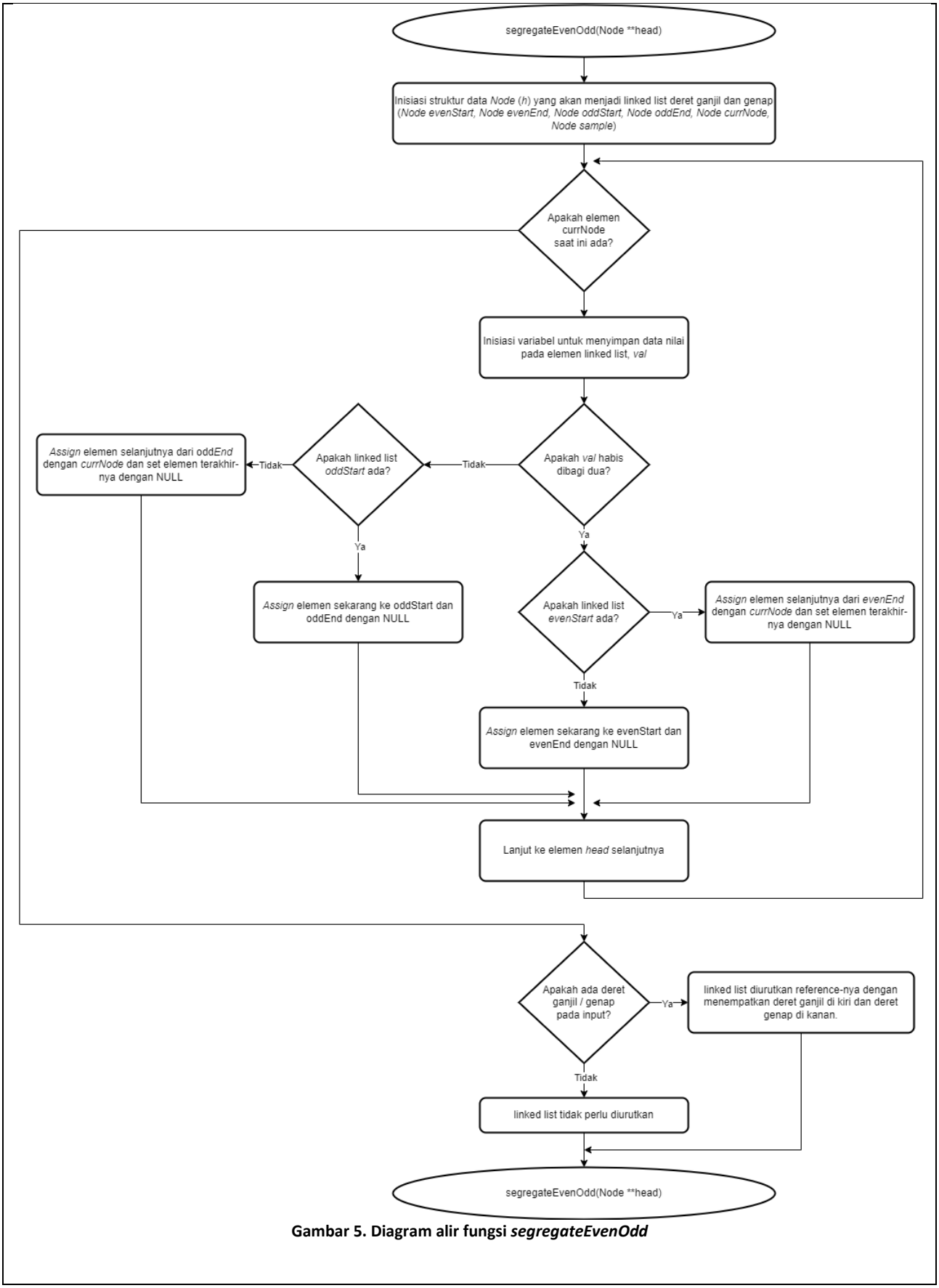
Gambar 2. Diagram alir fungsi *add*



Gambar 3. Diagram alir fungsi *swap*

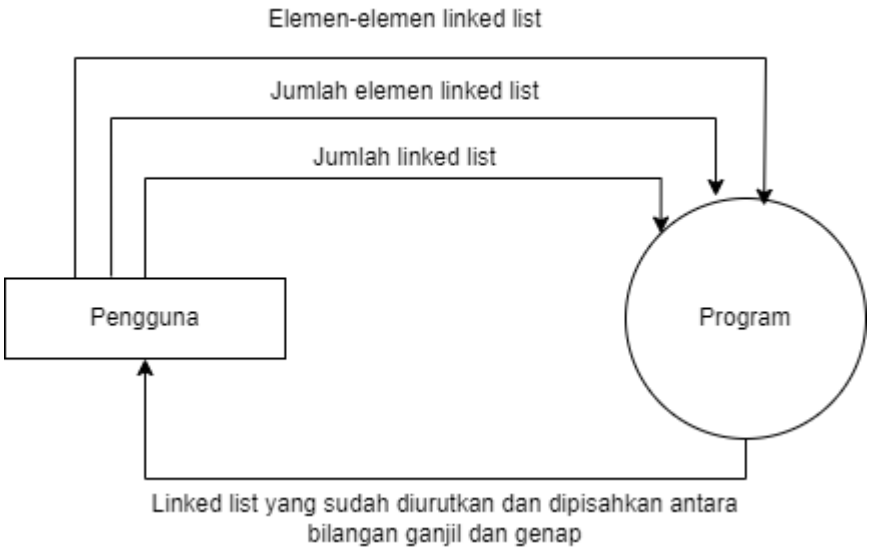


Gambar 4. Diagram alir fungsi *sort*

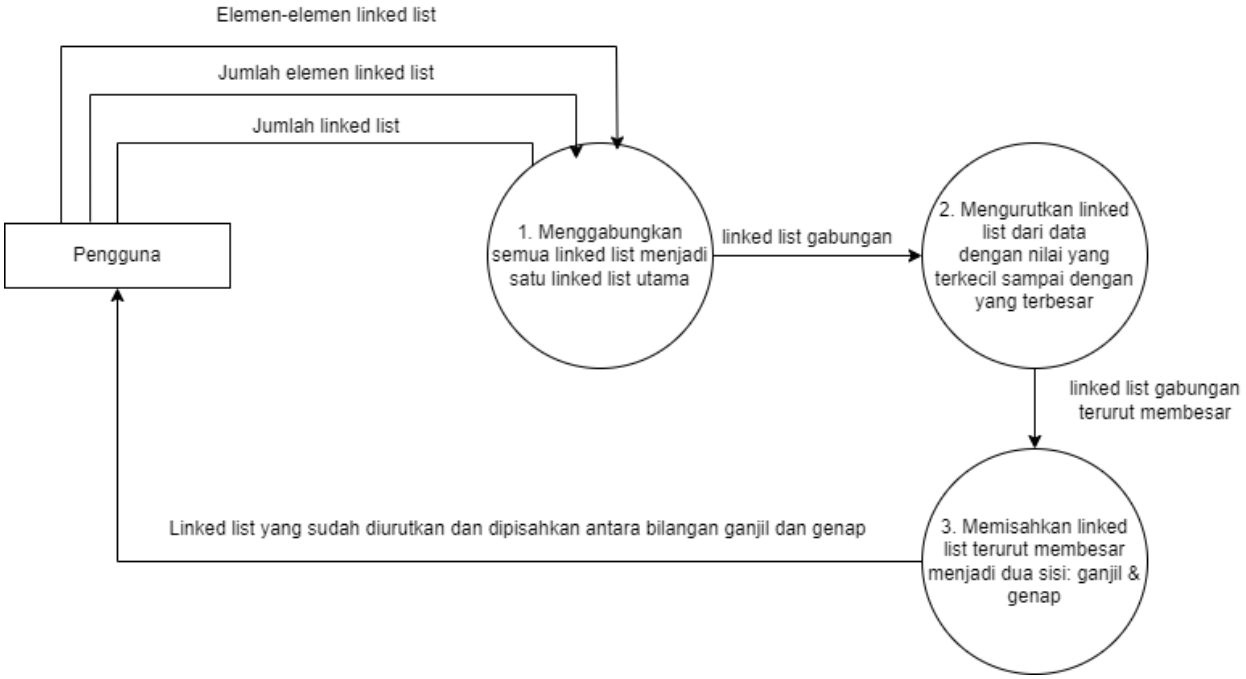


Gambar 5. Diagram alir fungsi *segregateEvenOdd*

Data Flow Diagram



Gambar 6. *Data flow diagram* level 0



Gambar 7. *Data flow diagram* level 1

Analisis Kompleksitas Waktu

Terdapat lima fungsi pada program ini dan masing fungsi memiliki kompleksitas waktunya sendiri-sendiri.
1.) fungsi *add(), segregateEvenOdd(),* dan *print()*:
O(n) karena fungsi melakukan pengulangan terhadap seluruh elemen linked list sekali.

2.) fungsi *sort():*
O(n^2) karena fungsi memerlukan *nested-loop* sebanyak dua kali untuk mengimplementasikan algoritma *sorting* menggunakan *bubble sort*.