

**LAPORAN PRAKTIKUM
ALGORITMA & STRUKTUR DATA
MODUL 3**



Single Linked List

Oleh:

Rizki Adhitiya Maulana

NIM. 2410817110014

**PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
MEI 2025**

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM ALGORITMA & STRUKTUR DATA
MODUL 3

Laporan Praktikum Algoritma & Struktur Data Modul 3 : Single Linked List ini disusun sebagai syarat lulus mata kuliah Praktikum Algoritma & Struktur Data. Laporan Praktikum ini dikerjakan oleh:

Nama Praktikan : Rizki Adhitiya Maulana
NIM : 2410817110014

Menyetujui,
Asisten Praktikum

Mengetahui,
Dosen Penanggung Jawab Praktikum

Muhammad Fauzan Ahsani
NIM. 2310817310009

Muti'a Maulida, S.Kom., M.TI.
NIP. 198810272019032013

DAFTAR ISI

LEMBAR PENGESAHAN	i
DAFTAR ISI.....	ii
DAFTAR TABEL.....	iv
DAFTAR GAMBAR	v
SOAL 1	1
A Output Program.....	12
B Pembahasan.....	18
SOAL 2	23
A Output Program.....	23
B Pembahasan.....	23
SOAL 3	24
A Output Program.....	24
B Pembahasan.....	24
SOAL 4	25
A Output Program.....	25
B Pembahasan.....	25
SOAL 5	26
A Output Program.....	26
B Pembahasan.....	26
SOAL 6	27
A Output Program.....	27
B Pembahasan.....	27
SOAL 7	28
A Output Program.....	28
B Pembahasan.....	28
SOAL 8	29
A Output Program.....	29

B Pembahasan.....	29
SOAL 9	30
A Pembahasan.....	30
SOAL 10	31
A Pembahasan.....	31
TAUTAN GIT HUB	32

DAFTAR TABEL

Tabel 1 Source Code Linked List	1
---------------------------------------	---

DAFTAR GAMBAR

Gambar 1 Tampilan Awal Program	12
Gambar 2 Tampilan Tambah Data dari Depan	12
Gambar 3 Tampilan Data Setelah Dilakukan Tambah Data Depan	12
Gambar 4 Tampilan Tambah Data dari Belakang.....	13
Gambar 5 Tampilan Data Setelah Dilakukan Tambah Data Belakang.....	13
Gambar 6 Tampilan Cari Data Yang Tidak Ada	13
Gambar 7 Tampilan Cari Data Yang Ada.....	14
Gambar 8 Tampilan Hapus Data Belakang.....	14
Gambar 9 Tampilan Data Setelah Dilakukan Hapus Data Belakang.....	14
Gambar 10 Tampilan Hapus Setiap Data Tertentu	15
Gambar 11 Tampilan Data Setelah Dilakukan Hapus Setiap Data Tertentu	15
Gambar 12 Tampilan Hapus Data Depan	15
Gambar 13 Tampilan Data Setelah Dilakukan Hapus Data Depan	16
Gambar 14 Tampilan Sisipkan Node Sebelum Data Tertentu	16
Gambar 15 Tampilan Data Setelah Dilakukan Sisipkan Node Sebelum	16
Gambar 16 Tampilan Sisipkan Node Setelah Data Tertentu	17
Gambar 17 Tampilan Data Setelah Dilakukan Sisipkan Node Setelah	17
Gambar 18 Tampilan Hapus Semua Elemen	17
Gambar 19 Tampilan Data Setelah Dilakukan Hapus Semua Elemen	18
Gambar 20 Tampilan Setelah Selesai Menyelesaikan Program.....	18
Gambar 21 Tampilan Nilai yang Diinput dari Depan.....	23
Gambar 22 Tampilan Nilai yang Diinput dari Belakang	24
Gambar 23 Pencarian Nilai 2 pada Linked List	25
Gambar 24 Pencarian Nilai 7 pada Linked List	26
Gambar 25 Menghapus Data dari Belakang Linked List.....	27
Gambar 26 Tampilan Linked List Setelah Dihapus	27
Gambar 27 Menghapus Setiap Nilai 3 yang Ada di Linked List	28
Gambar 28 Tampilan Linked List Setelah Setiap Nilai 3 Dihapus	28
Gambar 29 Tampilan Isi dari Linked List.....	29

SOAL 1

Cobalah program berikut, running, simpan program, dan screenshoot hasil running !

Tabel 1 Source Code Linked List

1	#include <conio.h>
2	#include <iostream>
3	#include <stdlib.h>
4	
5	using namespace std;
6	
7	typedef struct TNode
8	{
9	string data;
10	TNode *next;
11	};
12	
13	TNode *head, *tail;
14	
15	int pil;
16	char pilihan [2];
17	string dataBaru, dataDelete;
18	
19	void Init();
20	int Kosong();
21	
22	void TambahDepan();
23	void TambahBelakang();
24	void HapusDepan();
25	void HapusBelakang();
26	void Tampilkan();
27	void Reset();
28	void CariData();
29	void HapusData();
30	void SisipkanSebelum();
31	void SisipkanSetelah();
32	
33	int main()
34	{
35	do
36	{
37	cout<<"Single Linked List Circular (SLLC)"<<endl;
38	cout<<"===== "<<endl;
39	cout<<"1. Tambah Depan"<<endl;
40	cout<<"2. Tambah Belakang"<<endl;
41	cout<<"3. Hapus Depan"<<endl;
42	cout<<"4. Hapus Belakang"<<endl;
43	cout<<"5. Tampilkan Data"<<endl;
44	cout<<"6. Hapus Semua elemen"<<endl;

```

45         cout<<"7. Cari Data"<<endl;
46         cout<<"8. Hapus Setiap Data Tertentu"<<endl;
47         cout<<"9. Sisipkan Node/Data Baru Sebelum Data
Tertentu"<<endl;
48         cout<<"10. Sisipkan Node/Data Baru Setelah Data
Tertentu"<<endl;
49         cout<<"11. Keluar"<<endl;
50         cout<<"===== "<<endl;
51         cout<<"Pilihan : ";
52         cin>>pilihan;
53         pil = atoi(pilihan);
54
55         switch(pil)
56         {
57         case 1:
58             TambahDepan();
59             break;
60         case 2:
61             TambahBelakang();
62             break;
63         case 3:
64             HapusDepan();
65             cout<<"Data \""<<dataDelete<<"\" yang berada
di depan telah berhasil dihapus."<<endl;
66             break;
67         case 4:
68             HapusBelakang();
69             cout<<"Data \""<<dataDelete<<"\" yang berada
di belakang telah berhasil dihapus."<<endl;
70             break;
71         case 5:
72             Tampilkan();
73             break;
74         case 6:
75             Reset();
76             break;
77         case 7:
78             CariData();
79             break;
80         case 8:
81             HapusData();
82             break;
83         case 9:
84             SisipkanSebelum();
85             break;
86         case 10:
87             SisipkanSetelah();
88             break;

```



```

89         default:
90             cout<<"\nTERIMA KASIH"<<endl;
91             cout<<"Program was made by Rizki Adhitiya
Maulana (2410817110014)"<<endl;
92         }
93
94         cout<<"\nPress any key to continue!"<<endl;
95         getch();
96         system("cls");
97     }
98     while (pil<11);
99 }
100
101 void Init()
102 {
103     head = NULL;
104     tail = NULL;
105 }
106
107 int Kosong()
108 {
109     if(head == NULL)
110         return 1;
111     else
112         return 0;
113 }
114
115 void TambahDepan()
116 {
117     cout<<"Masukkan Data : ";
118     TNode *baru;
119     baru = new TNode;
120     cin>>dataBaru;
121     baru->data = dataBaru;
122     baru->next = baru;
123
124     if(Kosong() == 1)
125     {
126         head = baru;
127         tail = baru;
128     }
129     else
130     {
131         baru->next = head;
132         head = baru;
133         tail->next = head;
134     }
135     cout<<"Data  \\"<<dataBaru<<"\\"  telah  berhasil
dimasukkan di bagian depan."<<endl;

```

```

136     }
137
138 void TambahBelakang()
139 {
140     cout<<"Masukkan Data : ";
141     TNode *baru;
142     baru = new TNode;
143     cin>>dataBaru;
144     baru->data = dataBaru;
145     baru->next = baru;
146
147     if(Kosong() == 1)
148     {
149         head = baru;
150         tail = baru;
151     }
152     else
153     {
154         tail->next = baru;
155         tail = baru;
156         tail->next = head;
157     }
158     cout<<"Data  \""<<dataBaru<<"\"  telah  berhasil
dimasukkan di bagian belakang."<<endl;
159 }
160
161 void HapusDepan()
162 {
163     if(Kosong() == 0)
164     {
165         TNode *hapus;
166         hapus = head;
167         dataDelete = hapus->data;
168
169         if(head != tail)
170         {
171             head = head->next;
172             tail->next = head;
173         }
174         else
175         {
176             Init();
177         }
178
179         delete hapus;
180     }
181     else
182     {

```

```

183         cout<<"Tidak terdapat data pada Linked
List."<<endl;
184     }
185 }
186
187 void HapusBelakang()
188 {
189     if(Kosong() == 0)
190     {
191         TNode *hapus, *newTail;
192         hapus = tail;
193         dataDelete = hapus->data;
194
195         if(head != tail)
196         {
197             newTail = head;
198             while(newTail->next != tail)
199             {
200                 newTail = newTail->next;
201             }
202             tail = newTail;
203             tail->next = head;
204         }
205         else
206         {
207             Init();
208         }
209         delete hapus;
210     }
211     else
212     {
213         cout<<"Tidak terdapat data pada Linked
List."<<endl;
214     }
215 }
216
217 void Tampilkan()
218 {
219     if(Kosong() == 0)
220     {
221         TNode *bantu;
222         bantu = head;
223
224         do
225         {
226             cout<<bantu->data<<' ';
227             bantu = bantu->next;
228         }
229         while (bantu != head);

```

```

230         cout<<endl;
231     }
232     else
233     {
234         cout<<"Tidak terdapat data pada Linked
List."<<endl;
235     }
236 }
237
238 void Reset()
239 {
240     if(Kosong() == 0)
241     {
242         TNode *bantu, *hapus;
243         bantu = head;
244
245         do
246         {
247             hapus = bantu;
248             bantu = bantu->next;
249             delete hapus;
250         }
251         while (bantu != head);
252         Init();
253         cout<<"Seluruh elemen pada Linked List telah
dibersihkan."<<endl;
254     }
255     else
256     {
257         cout<<"Tidak terdapat data pada Linked
List."<<endl;
258     }
259 }
260
261 void CariData()
262 {
263     if(Kosong() == 0)
264     {
265         string cari;
266         cout<<"Masukkan data yang ingin dicari : ";
267         cin>>cari;
268
269         TNode *bantu, *hapus, *newTail, *bantuTampilkan;
270         bool apaDitemukan = false;
271
272         bantu = head;
273
274         do
275         {

```

```

276         if(cari == bantu->data)
277         {
278             cout<<"Setiap data yang berada di dalam
tanda kurung siku ([...]) "
279             <<"merupakan data yang anda
cari."<<endl;
280             cout<<"Linked List : ";
281             bantuTampilkan = head;
282
283             do
284             {
285                 if(cari == bantuTampilkan->data)
286                 {
287                     cout<<"["<<bantuTampilkan-
>data<<"] ";
288                 }
289                 else
290                 {
291                     cout<<bantuTampilkan->data<<' ';
292                 }
293                 bantuTampilkan = bantuTampilkan-
>next;
294             }
295             while (bantuTampilkan != head);
296
297             apaDitemukan = true;
298             cout<<endl;
299             break;
3003         }
01         bantu = bantu->next;
302     }
303     while (bantu != head);
304
305     if(apaDitemukan == false)
306     {
307         cout<<"Data \""<<cari<<"\" tidak ditemukan
pada Linked List."<<endl;
308     }
309     }
310     else
311     {
312         cout<<"Tidak terdapat data pada Linked
List."<<endl;
313     }
314 }
315
316 void HapusData()
317 {
318     if(Kosong() == 0)

```

```

319     {
320         string cari;
321         cout<<"Masukkan data yang ingin dihapus : ";
322         cin>>cari;
323
324         TNode *bantu, *sebelum, *hapus[255], *bantu2;
325         int hitung = 0;
326         bool apaDitemukan = false;
327
328         bantu = head;
329
330         do
331         {
332             bantu2 = bantu;
333             if(cari == bantu->data)
334             {
335                 hapus[hitung++] = bantu;
336                 apaDitemukan = true;
337                 if(bantu != head && bantu != tail)
338                 {
339                     sebelum->next = bantu->next;
340                     bantu2 = sebelum;
341                 }
342             }
343             sebelum = bantu2;
344             bantu = bantu->next;
345         }
346         while (bantu != head);
347
348         if(apaDitemukan == true)
349         {
350             for(int i=0; i<hitung; i++)
351             {
352                 if(hapus[i] == head)
353                 {
354                     HapusDepan();
355                 }
356                 else if(hapus[i] == tail)
357                 {
358                     HapusBelakang();
359                 }
360                 else
361                 {
362                     delete hapus[i];
363                 }
364             }
365             cout<<"Setiap data \""<<cari<<"\" yang
terdapat pada Linked List telah dihapus."<<endl;
366         }

```

```

367         else
368         {
369             cout<<"Data \""<<cari<<"\" tidak ditemukan
pada Linked List."<<endl;
370         }
371     }
372     else
373     {
374         cout<<"Tidak terdapat data pada Linked
List."<<endl;
375     }
376 }
377
378 void SisipkanSebelum()
379 {
380     if(Kosong() == 0)
381     {
382         TNode *bantu, *sebelum;
383         string nextData;
384         bool apaAda;
385
386         bantu = head;
387         sebelum = tail;
388
389         cout<<"Sisipkan data baru sebelum data : ";
390         cin>>nextData;
391
392         do
393         {
394             if(nextData == bantu->data)
395             {
396                 apaAda = true;
397                 break;
398             }
399             else
400             {
401                 sebelum = bantu;
402                 bantu = bantu->next;
403             }
404         }
405         while (bantu != head);
406
407         if(apaAda == true)
408         {
409             cout<<"Masukkan data yang ingin ditambahkan
: ";
410             cin>>dataBaru;
411
412             TNode *baru;

```

```

413         baru = new TNode;
414
415         baru->data = dataBaru;
416         baru->next = bantu;
417
418         sebelum->next = baru;
419
420         if(bantu == head)
421         {
422             head = baru;
423         }
424         cout<<"Data \""<<dataBaru<<"\" berhasil
disisipkan sebelum data \""<<nextData<<"\"."<<endl;
425     }
426     else
427     {
428         cout<<"Tidak terdapat data \""<<nextData<<"\"
pada Linked List."<<endl;
429     }
430 }
431 else
432 {
433     cout<<"Tidak terdapat data pada Linked
List."<<endl;
434 }
435 }
436
437 void SisipkanSetelah()
438 {
439     if(Kosong() == 0)
440     {
441         TNode *bantu;
442         string prevData;
443         bool apaAda;
444
445         bantu = head;
446
447         cout<<"Sisipkan data baru setelah data : ";
448         cin>>prevData;
449
450         do
451         {
452             if(prevData == bantu->data)
453             {
454                 apaAda = true;
455                 break;
456             }
457             else
458             {

```



```

459         bantu = bantu->next;
460     }
461 }
462 while (bantu != head);
463
464 if(apaAda == true)
465 {
466     cout<<"Masukkan data yang ingin ditambahkan
: ";
467     cin>>dataBaru;
468
469     TNode *baru;
470     baru = new TNode;
471
472     baru->data = dataBaru;
473     baru->next = bantu->next;
474
475     bantu->next = baru;
476
477     if(bantu == tail)
478     {
479         tail = baru;
480     }
481     cout<<"Data \""<<dataBaru<<"\" berhasil
disisipkan setelah data \""<<prevData<<"\"."<<endl;
482     }
483     else
484     {
485         cout<<"Tidak terdapat data \""<<prevData<<"\"
pada Linked List."<<endl;
486     }
487 }
488 else
489 {
490     cout<<"Tidak terdapat data pada Linked
List."<<endl;}
491 }

```

A Output Program

```

PS D:\task-3-single-circular-linked-list-Rizki-A-M> cd "d:\task-3-single-circular-linked-list-Rizki-A-M\" ; if ($?) { g++ SingleLinked
List.cpp -o SingleLinkedList } ; if ($?) { .\SingleLinkedList }
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan :

```

Gambar 1 Tampilan Awal Program

```

PS D:\task-3-single-circular-linked-list-Rizki-A-M> cd "d:\task-3-single-circular-linked-list-Rizki-A-M\" ; if ($?) { g++ SingleLinked
List.cpp -o SingleLinkedList } ; if ($?) { .\SingleLinkedList }
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 1
Masukkan Data : 3
Data "3" telah berhasil dimasukkan di bagian depan.
Press any key to continue!

```

Gambar 2 Tampilan Tambah Data dari Depan

```

Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 5
12 10 9 7 4 3
Press any key to continue!

```

Gambar 3 Tampilan Data Setelah Dilakukan Tambah Data Depan

```

task-3-single-circular-linked-list-Rizki-A-M
File Edit Selection View ... task-3-single-circular-linked-list-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 2
Masukkan Data : 3
Data -> 3 telah berhasil dimasukkan di bagian belakang.
Press any key to continue!

```

Gambar 4 Tampilan Tambah Data dari Belakang

```

task-3-single-circular-linked-list-Rizki-A-M
File Edit Selection View ... task-3-single-circular-linked-list-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 5
12 10 9 7 4 3 3 7 1 4 3
Press any key to continue!

```

Gambar 5 Tampilan Data Setelah Dilakukan Tambah Data Belakang

```

task-3-single-circular-linked-list-Rizki-A-M
File Edit Selection View ... task-3-single-circular-linked-list-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 7
Masukkan data yang ingin dicari : 2
Data "2" tidak ditemukan pada Linked List.
Press any key to continue!

```

Gambar 6 Tampilan Cari Data Yang Tidak Ada

```

task-3-single-circular-linked-list-Rizki-A-M
File Edit Selection View ... task-3-single-circular-linked-list-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 7
Masukkan data yang ingin dicari : 7
Setiap data yang berada di dalam tanda kurung siku ([...]) merupakan data yang anda cari.
Linked List : 12 10 9 [7] 4 3 3 [7] 1 4 3

Press any key to continue!

```

Gambar 7 Tampilan Cari Data Yang Ada

```

task-3-single-circular-linked-list-Rizki-A-M
File Edit Selection View ... task-3-single-circular-linked-list-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 4
Data "4" yang berada di belakang telah berhasil dihapus.

Press any key to continue!

```

Gambar 8 Tampilan Hapus Data Belakang

```

task-3-single-circular-linked-list-Rizki-A-M
File Edit Selection View ... task-3-single-circular-linked-list-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 5
12 10 9 7 4 3 3 7 1 4

Press any key to continue!

```

Gambar 9 Tampilan Data Setelah Dilakukan Hapus Data Belakang

```

Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 8
Masukkan data yang ingin dihapus : 3
Setiap data "3" yang terdapat pada Linked List telah dihapus.

Press any key to continue!

```

Gambar 10 Tampilan Hapus Setiap Data Tertentu

```

Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 5
12 10 9 7 4 7 1 4

Press any key to continue!

```

Gambar 11 Tampilan Data Setelah Dilakukan Hapus Setiap Data Tertentu

```

Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 3
Data "12" yang berada di depan telah berhasil dihapus.

Press any key to continue!

```

Gambar 12 Tampilan Hapus Data Depan

```

task-3-single-circular-linked-list-Riki-A-M
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 5
10 9 7 4 7 1 4

Press any key to continue!

```

Gambar 13 Tampilan Data Setelah Dilakukan Hapus Data Depan

```

task-3-single-circular-linked-list-Riki-A-M
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 9
Sisipkan data baru sebelum data : 9
Masukkan data yang ingin ditambahkan : 1
Data "1" berhasil disisipkan sebelum data "9".

Press any key to continue!

```

Gambar 14 Tampilan Sisipkan Node Sebelum Data Tertentu

```

task-3-single-circular-linked-list-Riki-A-M
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 5
10 1 9 7 4 7 1 4

Press any key to continue!

```

Gambar 15 Tampilan Data Setelah Dilakukan Sisipkan Node Sebelum

```

task-3-single-circular-linked-list-Riki-A-M
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 10
Sisipkan data baru setelah data : 1
Masukkan data yang ingin ditambahkan : 8
Data "8" berhasil disisipkan setelah data "1".

Press any key to continue!

```

Gambar 16 Tampilan Sisipkan Node Setelah Data Tertentu

```

task-3-single-circular-linked-list-Riki-A-M
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 5
10 1 8 9 7 4 7 1 4

Press any key to continue!

```

Gambar 17 Tampilan Data Setelah Dilakukan Sisipkan Node Setelah

```

task-3-single-circular-linked-list-Riki-A-M
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 6
Seluruh elemen pada Linked List telah dibersihkan.

Press any key to continue!

```

Gambar 18 Tampilan Hapus Semua Elemen

```

task-3-single-circular-linked-list-Rizki-A-M
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 5
Tidak terdapat data pada Linked List.

Press any key to continue!

```

Gambar 19 Tampilan Data Setelah Dilakukan Hapus Semua Elemen

```

task-3-single-circular-linked-list-Rizki-A-M
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 11

TERIMA KASIH
Program was made by Rizki Adhitiya Maulana (2410817110014)

Press any key to continue!

```

Gambar 20 Tampilan Setelah Selesai Menyelesaikan Program

B Pembahasan

Pada baris [1] sampai [3] terdapat `#include` yang mana digunakan untuk mengakses sebuah file yang diinginkan. `<iostream>` yang ada digunakan untuk input dan output. Kemudian `<conio.h>` digunakan agar menyediakan fungsi fungsi yang berguna ketika ada interaksi langsung dengan keyboard, tanpa perlu menekan Enter. Terus `<stdlib.h>` digunakan untuk fungsi fungsi manajemen memori, konversi angka, kontrol proses, dan lingkungan program.

Pada baris [5] terdapat using `namespace std;` yang mana digunakan untuk menghindari penulisan `std.`

Pada baris [7] sampai [11] terdapat `struct TNode` yang mana digunakan untuk menyimpan elemen-elemen dari linked list, dimana variabel `string data` digunakan

untuk menyimpan isi atau informasi dari node tersebut seperti angka yang di input ke dalam list. Kemudian, *TNode *next* digunakan untuk menunjuk ke node berikutnya dalam urutan linked list, node terakhir yang ada dalam urutan linked list akan menunjuk kembali ke node pertama.

Pada baris [13] terdapat *TNode *head, *tail;* yang mana *TNode *head* menunjuk pointer node pertama dan *TNode *tail* menunjuk pointer node terakhir.

Pada baris [15] sampai [17] terdapat *int pil* yang mana digunakan untuk menyimpan variabel Integer atau bilangan bulat. Terus *char pilihan [2]* yang mana digunakan untuk menyimpan variabel character, ditambah array sebagai batasan input dari user, *string dataBaru* dan *dataDelete* yang digunakan untuk menyimpan variabel string atau katakter.

Pada baris [19] sampai [31] terdapat penamaan fungsi yang akan dimasukkan ke dalam program Linked list.

Pada baris [33] sampai [99] terdapat *int main()* yang mana digunakan untuk menjalankan dan menampilkan menu SLLC. Menu yang akan ditampilkan di dalam sistem ada sebanyak 11 buah, terdiri dari tambah depan dan tambah belakang, hapus depan dan hapus belakang, tampilkan data, sapus semua elemen, cari data, hapus data tertentu, sisipkan data sebelum data tertentu dan sisipkan data setelah data tertentu. Setiap pilihan yang ada akan menampilkan tampilan berbeda sesuai dengan fungsi yang ada di dalam *switch case* yang dimasukkan pada program yang akan dijalankan. Terdapat *getch()* untuk menunggu tombol yang ditekan oleh pengguna dan membersihkan layar menggunakan *system("cls")*. Terus program akan terus berjalan selama user tidak memilih pilihan sebelas (11) untuk keluar atau menghentikan program yang ada.

Pada baris [101] sampai [105] terdapat *void Init()* yang mana digunakan untuk menginisialisasikan kondisi awal dari linked list circular. Pointer atau variabel head akan disetting atau diatur dalam keadaan NULL, begitu juga dengan pointer atau variabel tail akan disetting atau diatur dalam keadaan NULL.

Pada baris [107] sampai [113] terdapat *int Kosong()* yang mana digunakan untuk melakukan pengecekan pada linked list, apakah dalam keadaan kosong atau

tidak. Fungsi ini akan mengembalikan nilai 1 apabila linked list dalam keadaan kosong dan 0 apabila tidak dalam keadaan kosong. Fungsi ini akan dipanggil pada saat melakukan operasi penambahan, penghapusan dan menampilkan.

Pada baris [115] sampai [136] terdapat *void TambahDepan()* yang mana digunakan untuk menambahkan node baru ke bagian depan dari linked list. Apabila linked list dalam keadaan kosong, node yang baru ditambahkan akan menjadi head dan tail. Kemudian, apabila linked list dalam keadaan tidak kosong, node yang baru ditambahkan akan menjadi head baru dan node sebelumnya yang jadi head dan tail dalam satu waktu akan menjadi tail.

Pada baris [138] sampai [159] terdapat *void TambahBelakang()* yang mana digunakan untuk menambahkan node baru ke bagian belakang dari linked list. Apabila linked list dalam keadaan kosong, node yang baru ditambahkan akan menjadi tail dan head. Kemudian, apabila linked list dalam keadaan tidak kosong, node yang baru ditambahkan akan menjadi tail baru dan node sebelumnya yang jadi tail dan head dalam satu waktu akan menjadi head.

Pada baris [161] sampai [185] terdapat *void HapusDepan()* yang mana digunakan untuk menghapus node pertama yang terdapat pada linked list. Apabila terdapat satu node saja pada linked list, maka setelah dilakukan hapus depan linked list akan diatur menjadi kosong atau NULL menggunakan fungsi *Init()*. Kemudian, apabila terdapat lebih dari satu node yang terdapat pada linked list, node pertama yang dihapus akan digantikan dengan elemen yang ada di setelah sebagai head terbaru dan node pertama baru.

Pada baris [187] sampai [215] terdapat *void HapusBelakang()* yang mana digunakan untuk menghapus node terakhir yang terdapat pada linked list. Apabila terdapat satu node saja pada linked list, maka setelah dilakukan hapus belakang linked list akan diatur menjadi kosong atau NULL menggunakan fungsi *Init()*. Kemudian, apabila terdapat lebih dari satu node yang terdapat pada linked list, node terakhir yang dihapus akan digantikan dengan elemen yang ada di sebelumnya sebagai tail terbaru dan node terakhir baru.

Pada baris [217] sampai [236] terdapat *void Tampilkan()* yang mana digunakan untuk menampilkan seluruh isi linked list yang ada, dimulai dari node pertama hingga ke node terakhir. Walaupun hanya terdapat satu node di dalam linked list, program akan tetap mencetaknya dan akan berhenti ketika semua elemen yang ada di dalam linked list ditampilkan semua.

Pada baris [238] sampai [259] terdapat *void Reset()* yang mana digunakan untuk menghapus semua node yang ada pada linked list, baik dari node pertama hingga node terakhir. Setelah di lakukan penghapusan untuk semua node yang ada di dalam linked list, kemudian akan dipanggil fungsi *Init()* untuk mengatur linked list ke dalam kondisi kosong.

Pada baris [261] sampai [314] terdapat *void CariData()* yang mana digunakan untuk mencari data tertentu yang ada di dalam linked list. Apabila ditemukan kesamaan data yang ada di dalam tanda kurung siku dengan yang ada di dalam linked list, maka data yang ada di dalam linked list akan ikut di cetak dalam tanda kurung siku. Namun, apabila tidak ditemukan kesamaan antara data yang ada di dalam tanda kurung siku dengan yang ada di dalam linked list, maka akan muncul tampilan pesan kepada pengguna kalo data A tidak ditemukan.

Pada baris [316] sampai [376] terdapat *void HapusData()* yang mana digunakan untuk menghapus semua node yang ada di dalam linked list, berdasarkan data yang di inputkan oleh pengguna. Fungsi ini akan menghapus node tertentu di dalam linked list ketika pengguna memasukkan sebuah data atau nilai yang ingin di hapus. Ketika selesai memasukkan data yang diinginkan, sistem akan menghapus setiap node yang memiliki kesamaan dengan nilai yang dimasukkan hingga tidak terdapatnya data tersebut di dalam linked list.

Pada baris [378] sampai [435] terdapat *void SisipkanSebelum()* yang mana digunakan untuk menyisipkan atau memasukkan node baru sebelum node tertentu, sesuai dengan data yang dimasukkan oleh pengguna sebagai data yang akan disisipkan dan data target untuk penyisipan. Sebelum dilakukan penyisipan, terlebih dahulu dilakukan penelusuran pada linked list dari head menuju tail untuk menemukan sebuah node yang menjadi target untuk disisipkan. Setelah target

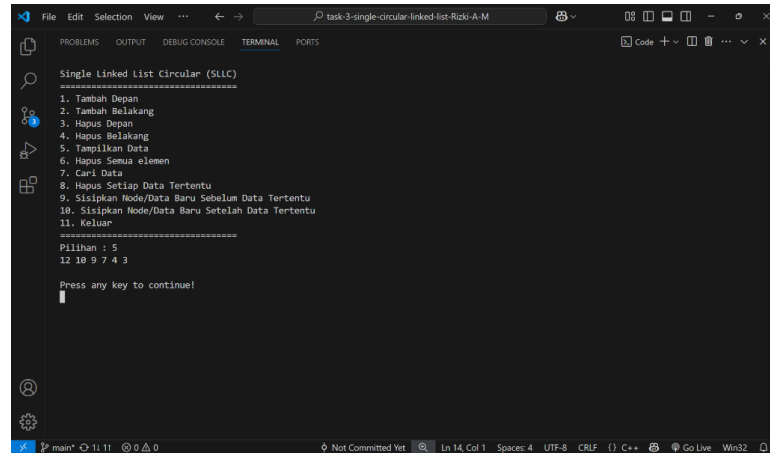
untuk penyisipan ditemukan, maka akan ditambahkan atau disisipkan sebuah node di depannya atau sebelum node tersebut. Apabila node yang menjadi target sisipkan sebelum adalah head dalam linked list maka node yang baru di tambahkan akan menjadi head yang baru.

Pada baris [437] sampai [491] terdapat *void SisipkanSesudah()* yang mana digunakan untuk menyisipkan atau memasukkan node baru setelah node tertentu sesuai dengan data yang dimasukkan oleh pengguna sebagai data yang akan disisipkan dan data target untuk penyisipan. Sebelum dilakukan penyisipan, terlebih dahulu dilakukan penelusuran pada linked list dari head menuju tail untuk menemukan sebuah node yang menjadi target untuk disisipkan. Setelah target untuk penyisipan ditemukan, maka akan ditambahkan atau disisipkan sebuah node di belakang atau setelah node tersebut. Apabila node yang menjadi target sisipkan setelah adalah tail dalam linked list maka node yang baru di tambahkan akan menjadi tail yang baru.

SOAL 2

Lakukan tambah data depan 3, 4, 7, 9, 10, 12 dan kemudian lakukan tampilkan data lalu screenshoot hasilnya !

A Output Program



```
task-3-single-circular-linked-list-Rizki-A-M
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 5
12 10 9 7 4 3

Press any key to continue!
```

Gambar 21 Tampilan Nilai yang Diinput dari Depan

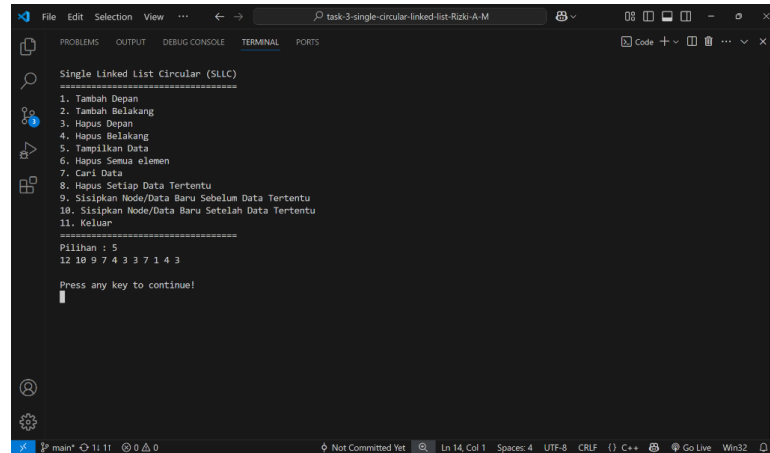
B Pembahasan

Ketika data dimasukkan ke dalam Linked List melalui bagian depan, dengan urutan masuk mulai dari 3, 4, 7, 9, 10, dan 12. Angka 3 yang pertama kali dimasukkan akan muncul di sebelah kiri ketika di tampilkan. Data-data yang muncul setelah data pertama ditambahkan akan menjadi head dari Linked List dan data sebelumnya akan digeser ke samping kiri atau ke belakang.

SOAL 3

Lakukan tambah data belakang 3, 7, 1, 4, 3 dan kemudian lakukan tampilkan data lalu screenshoot hasilnya !

A Output Program



```
task-3-single-circular-linked-list-Rizki-A-M

Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 5
12 10 9 7 4 3 3 7 1 4 3

Press any key to continue!
```

Gambar 22 Tampilan Nilai yang Diinput dari Belakang

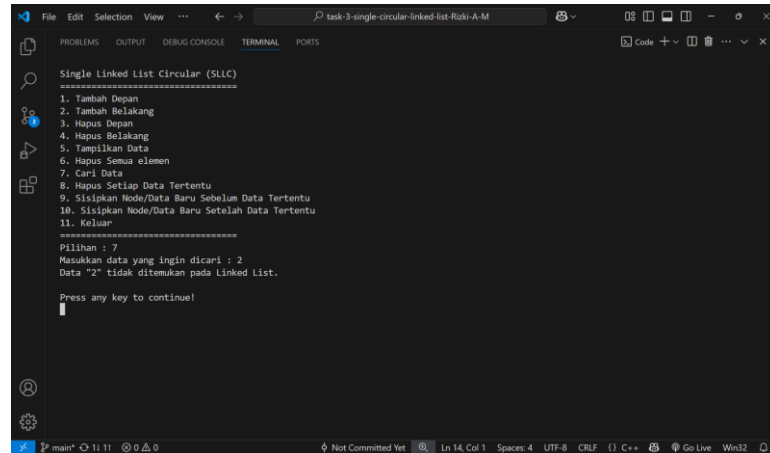
B Pembahasan

Ketika data dimasukkan ke dalam Linked List melalui bagian belakang, dengan urutan masuk mulai dari 3, 7, 1, 4, dan 3. Angka 3 yang pertama kali dimasukkan akan muncul di sebelah kiri ketika di tampilkan. Data-data yang muncul setelah data pertama ditambahkan akan menjadi tail dari Linked List dan tampilkan dari Linked List akan sama seperti saat kita memasukkan data ke dalam Linked List.

SOAL 4

Apa yang terjadi jika mencari angka 2 pada Single Linked List Circular (SLLC) pada data yang telah ditambahkan/dimasukkan sebelumnya dan screenshoot hasilnya

A Output Program



```
task-3-single-circular-linked-list-Riki-A-M

Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 7
Masukkan data yang ingin dicari : 2
Data "2" tidak ditemukan pada Linked List.
Press any key to continue!
```

Gambar 23 Pencarian Nilai 2 pada Linked List

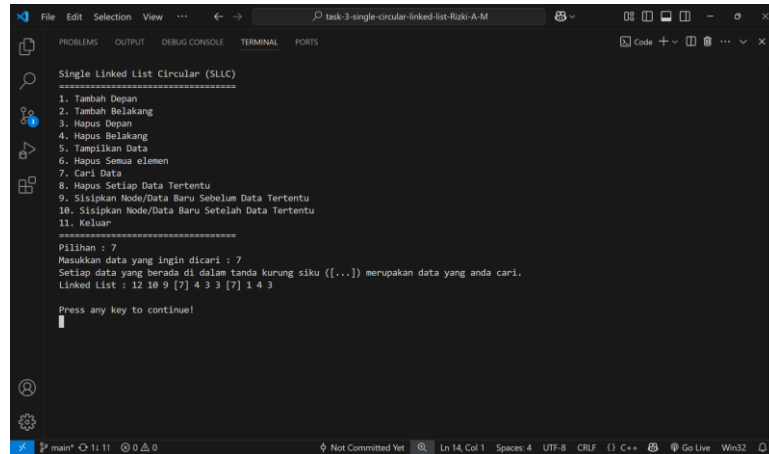
B Pembahasan

Berdasarkan data yang telah dimasukkan pada soal 2 dan soal 3, ketika dilakukan sebuah pencarian untuk nilai 2. Sistem akan memberikan informasi bahwa nilai tersebut tidak terdapat atau tidak ditemukan pada Linked List.

SOAL 5

Coba cari angka 7 dan screenshoot hasilnya !

A Output Program



```
task-3-single-circular-linked-list-Rizi-A-M
File Edit Selection View ... < ->
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 7
Masukkan data yang ingin dicari : 7
Setiap data yang berada di dalam tanda kurung siku ([...]) merupakan data yang anda cari.
Linked List : 12 10 9 [7] 4 3 3 [7] 1 4 3
Press any key to continue!
```

Gambar 24 Pencarian Nilai 7 pada Linked List

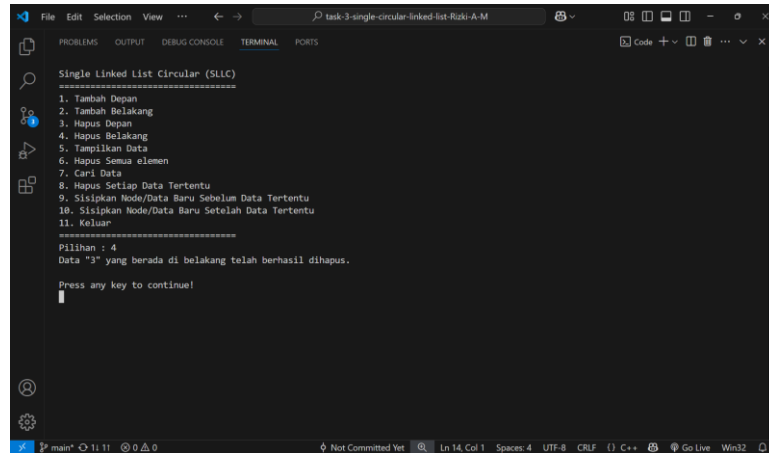
B Pembahasan

Berdasarkan data yang telah dimasukkan pada soal 2 dan soal 3, ketika dilakukan sebuah pencarian untuk nilai 7. Sistem akan memberikan informasi bahwa nilai yang dicari akan ditandai dengan kurung siku [...] pada Linked List.

SOAL 6

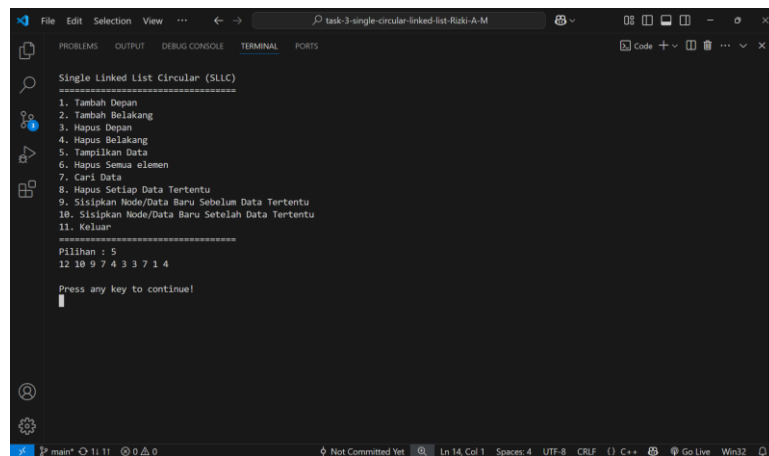
Lakukan hapus belakang dan kemudian lakukan tampilkan data lalu screenshoot hasilnya !

A Output Program



```
File Edit Selection View ... task-3-single-circular-linked-list-Riki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 4
Data : 3
Data 3 yang berada di belakang telah berhasil dihapus.
Press any key to continue!
```

Gambar 25 Menghapus Data dari Belakang Linked List



```
File Edit Selection View ... task-3-single-circular-linked-list-Riki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 5
12 10 9 7 4 3 3 7 1 4
Press any key to continue!
```

Gambar 26 Tampilan Linked List Setelah Dihapus

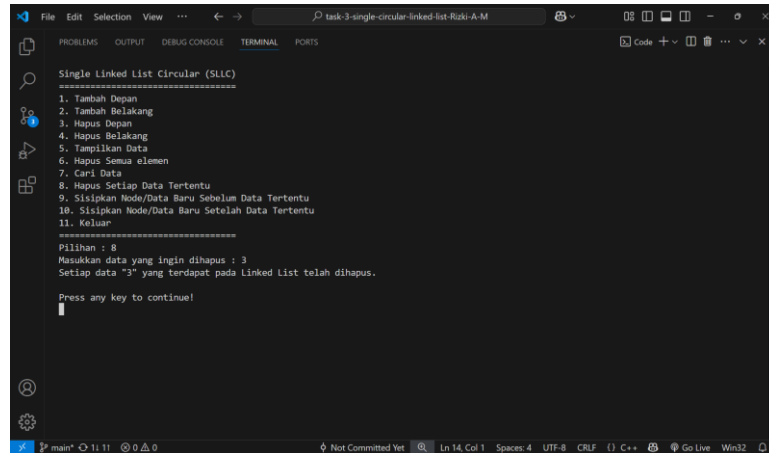
B Pembahasan

Berdasarkan data yang telah dimasukkan pada soal 2 dan soal 3, ketika dilakukan sebuah penghapusan data untuk bagian belakang atau tail. Nilai 3 yang terletak di bagian paling belakang pada Linked List (tail) akan dihapuskan dan ketika Linked List di tampilkan nilai yang ada disebelah kiri dari nilai yang dihapus akan menjadi tail yang baru.

SOAL 7

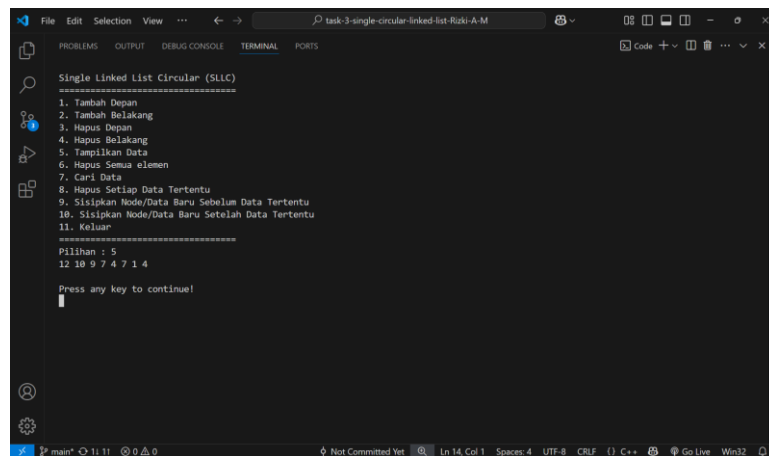
Lakukan hapus setiap angka 3 dan kemudian lakukan tampilkan data lalu screenshoot hasilnya !

A Output Program



```
File Edit Selection View ... task-3-single-circular-linked-list-Riki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 8
Masukkan data yang ingin dihapus : 3
Setiap data "3" yang terdapat pada Linked List telah dihapus.
Press any key to continue!
```

Gambar 27 Menghapus Setiap Nilai 3 yang Ada di Linked List



```
File Edit Selection View ... task-3-single-circular-linked-list-Riki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 5
12 18 9 7 4 7 1 4
Press any key to continue!
```

Gambar 28 Tampilan Linked List Setelah Setiap Nilai 3 Dihapus

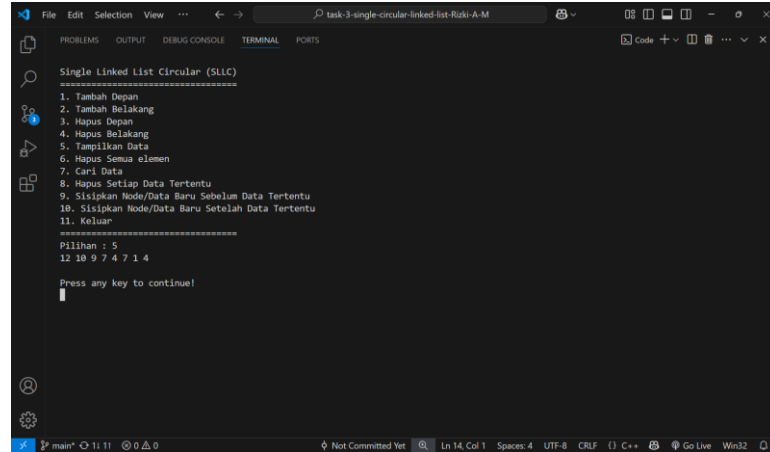
B Pembahasan

Berdasarkan data yang telah dimasukkan pada soal 2 dan soal 3, ketika dilakukan sebuah penghapusan untuk setiap Nilai 3 yang ada pada Linked List. Nilai 3 yang terdapat pada Linked List akan dihapuskan semua dan ketika Linked List di tampilkan tidak akan terdapat nilai 3 di dalamnya.

SOAL 8

Tampilkan data lalu jelaskan yang mana head dan yang mana tail.

A Output Program



```
task-3-single-circular-linked-list-Rizi-A-M
File Edit Selection View ... task-3-single-circular-linked-list-Rizi-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 5
12 10 9 7 4 7 1 4
Press any key to continue!
```

Gambar 29 Tampilan Isi dari Linked List

B Pembahasan

Berdasarkan data yang telah dimasukkan pada soal 2 dan soal 3, kemudian dilakukannya penghapusan pada soal 6 dan soal 7. Data yang berada di paling depan atau di samping kiri akan menjadi head yaitu 12, dapat juga dibuktikan dengan melakukan sebuah operasi penambahan atau penghapusan data dari depan yang mana akan mempengaruhi data atau nilai dari 12 itu sendiri. Apabila ditambah data 1 pada Linked List dari depan data yang baru ditambahkan tersebut yang akan menjadi head. Begitu juga dengan penghapusan data atau nilai dari 12, maka data atau nilai yang ada di belakangnya yang akan menjadi head baru. Sedangkan data yang berada di paling belakang atau di samping kanan akan menjadi tail yaitu 4, dapat juga dibuktikan dengan melakukan sebuah operasi penambahan atau penghapusan data dari belakang yang mana akan mempengaruhi data atau nilai dari 4 itu sendiri. Apabila ditambah data 1 pada Linked List dari belakang data yang baru ditambahkan tersebut yang akan menjadi tail. Begitu juga dengan penghapusan data atau nilai dari 4, maka data atau nilai yang ada di depannya yang akan menjadi tail baru.

SOAL 9

Jika baris ke 103 dan 104 dihapus maka apa yang akan terjadi pada saat memasukkan data, dan jelaskan mengapa?

A Pembahasan

Apabila baris ke-103 dan 104 dihapuskan dari program yang mana pada baris itu terdapat *if(head == NULL) return 1;* dan *else return 0;* di dalamnya. Maka akan fungsi pengecekan untuk mengetahui apakah Linked List kosong atau berisi tidak bisa dilakukan di dalam program. Tanpa kedua baris kode tersebut, program masih bisa berjalan, namun berisiko mengalami kesalahan atau perilaku yang tidak terduga, seperti penggunaan garbage value atau pengaksesan memori yang tidak valid, yang dapat menyebabkan crash atau error lainnya.

SOAL 10

Jelaskan apa itu variabel head dan tail pada SLLC!

A Pembahasan

Pada Single Linked List Circular (SLLC), variabel head merupakan pointer yang menunjuk ke node pertama dalam Linked List, atau dengan kata lain, variabel head berfungsi sebagai tempat pertama data akan ditelusuri. Sedangkan variabel tail merupakan pointer yang menunjuk ke node terakhir dalam Linked List, atau tempat terakhir data akan ditelusuri.

Karena SLLC bersifat melingkar (*circular*), maka node terakhir (*tail*) akan menunjuk kembali ke node pertama (*head*), yang mana akan menjadikannya sebuah struktur yang terus berputar. Singkatnya, tail sebagai node terakhir akan menunjuk kembali ke head, sehingga memungkinkan traversal list secara terus-menerus tanpa ada akhir.

TAUTAN GIT HUB

[https://github.com/Rizki-A-M/Rizki-A-M-
PRAKTIKUM_ALGORITMA_DAN_STRUKTUR_DATA.git](https://github.com/Rizki-A-M/Rizki-A-M-PRAKTIKUM_ALGORITMA_DAN_STRUKTUR_DATA.git)