

**Started on** Monday, 7 November 2022, 11:02 AM

**State** Finished

**Completed on** Monday, 7 November 2022, 11:10 AM

**Time taken** 8 mins 44 secs

**Grade** 600.00 out of 600.00 (100%)

Question **1**  
Correct  
Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Buatlah implementasi listlinier.c dari file header [listlinier.h](#)!

C

 [listlinier.c](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.00 sec, 1.60 MB
2	10	Accepted	0.00 sec, 1.60 MB
3	10	Accepted	0.00 sec, 1.71 MB
4	10	Accepted	0.00 sec, 1.65 MB
5	10	Accepted	0.00 sec, 1.57 MB
6	10	Accepted	0.00 sec, 1.61 MB
7	10	Accepted	0.00 sec, 1.61 MB
8	10	Accepted	0.00 sec, 1.64 MB
9	10	Accepted	0.00 sec, 1.64 MB
10	10	Accepted	0.00 sec, 1.66 MB

Question **2**

Correct

Mark 100.00 out of 100.00

Time limit

1 s

Memory limit

64 MB

Dengan menggunakan ADT List Linier dengan representasi fisik pointer yang Anda buat pada soal 1, buatlah program yang mengelola statistik daftar nilai sebuah mata kuliah (asumsikan nilai selalu 0..100) sebagai berikut:

1. Membuat sebuah list kosong untuk menyimpan daftar nilai mata kuliah.
2. Untuk setiap list: Membaca sekumpulan nilai integer dari keyboard sampai pengguna memasukkan angka di luar range nilai (di luar 0..100). Setiap elemen ditambahkan sebagai elemen pertama.
3. Jika list tidak kosong, menuliskan:
  - ☐ Ada berapa banyak nilai yang ada di daftar
  - ☐ Nilai tertinggi yang ada di daftar.
  - ☐ Nilai terendah yang ada di daftar.
  - ☐ Nilai rata-rata (presisi: 2 digit di belakang koma).
  - ☐ Mengkopi isi daftar ke sebuah list yang lain dan membalik isinya dan menuliskan isinya.
  - ☐ Menuliskan isi daftar yang asli (sebelum dibalik).
4. Jika list kosong, menuliskan "Daftar nilai kosong".

Berikut adalah contoh input/output:

Input	Output
100	6
98	
78	
100	
76	
76	90.17
89	[100,98,78,100,76,89]
-1	[89,76,100,78,98,100]
-999	Daftar nilai kosong

C

 [soal2.c](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	20	Accepted	0.00 sec, 1.70 MB
2	20	Accepted	0.00 sec, 1.58 MB
3	20	Accepted	0.00 sec, 1.59 MB
4	20	Accepted	0.01 sec, 1.50 MB
5	20	Accepted	0.00 sec, 1.57 MB

Question **3**

Correct

Mark 100.00 out of 100.00

Time limit

1 s

Memory limit

64 MB

Implementasikan fungsi OddEvenList yang pada pada file [OddEvenList.h](#) . Submit file dengan nama **OddEvenList.c!**

Gunakan ADT list linier yang dibuat pada nomor 1.

C

⚙ [OddEvenList.c](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	12	Accepted	0.00 sec, 1.55 MB
2	12	Accepted	0.00 sec, 1.66 MB
3	12	Accepted	0.00 sec, 1.64 MB
4	12	Accepted	0.00 sec, 1.60 MB
5	12	Accepted	0.00 sec, 1.50 MB
6	12	Accepted	0.00 sec, 1.66 MB
7	12	Accepted	0.00 sec, 1.59 MB
8	16	Accepted	0.00 sec, 1.52 MB

Question **4**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Pada sistem operasi, akses memori pada RAM cukup lambat. Karena itu, digunakan cache yang jauh lebih cepat. Namun cache berukuran sangat kecil, jadi cache hanya menyimpan sedikit bagian dari RAM. Karena tidak semua nilai disimpan, saat dibutuhkan sebuah nilai  $x$ , cache bisa hit (cache memiliki nilai  $x$ ) atau miss (cache tidak memiliki nilai  $x$ ).

Salah satu implementasi cache adalah dengan skema LRU, yakni Least Recently Used. Pada skema ini, apabila nilai  $x$  tidak ada di cache, cache akan menghapus nilai di cache yang paling lama sudah tidak digunakan. Jadi, cache menyimpan daftar nilai, terurut dari yang paling baru digunakan, sampai yang paling lama digunakan. Cache dapat direpresentasikan sebagai sebuah list linier. Cache akan diinisialisasi dengan nilai 1 sampai  $N$ . Lalu, akan ada  $Q$  buah operasi pengambilan nilai  $x$ :

- Jika  $x$  ada di cache, nilai  $x$  dipindah ke depan cache.
- Jika  $x$  tidak ada di cache, nilai paling akhir dihapus dari cache dan  $x$  dimasukkan ke depan cache.
- Untuk setiap operasi, tuliskan apakah operasi "hit" atau "miss". Lalu, tuliskan isi list.

Gunakan [listlinier.h](#), [listliner.c](#), dan kumpulkan **cache.c**

Contoh input / output:

Input	Output	Penjelasan
	hit [4,1,2,3,5,6]	$N = 6$
6	hit [6,4,1,2,3,5]	$Q = 5$
5	hit [6,4,1,2,3,5]	Pada awalnya, cache berisi [1,2,3,4,5,6].
4	miss	Ada 5 operasi yang dilakukan.
6	[7,6,4,1,2,3]	Pada operasi pertama, 4 ada di cache.
7	miss	Pada operasi kedua, 6 juga ada di cache.
5	[5,7,6,4,1,2]	Pada operasi ketiga, 7 tidak ada di cache, jadi 5 dihapus dan 7 ditambahkan ke depan cache.
5	hit [5,7,6,4,1,2]	Pada operasi keempat, 5 sudah tidak ada di cache, jadi 3 dihapus dan 5 ditambahkan ke depan cache.
	hit [5,7,6,4,1,2]	Pada operasi kelima, 5 sudah ada di cache.

C

 [cache.c](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	7	Accepted	0.00 sec, 1.64 MB
2	7	Accepted	0.00 sec, 1.62 MB
3	7	Accepted	0.00 sec, 1.71 MB
4	7	Accepted	0.00 sec, 1.59 MB
5	7	Accepted	0.00 sec, 1.67 MB
6	7	Accepted	0.00 sec, 1.61 MB
7	7	Accepted	0.00 sec, 1.50 MB

7	7	Accepted	0.00 sec, 1.50 MB
No	Score	Verdict	Description

8	7	Accepted	0.00 sec, 1.65 MB
9	7	Accepted	0.00 sec, 1.64 MB
10	7	Accepted	0.00 sec, 1.64 MB
11	7	Accepted	0.00 sec, 1.66 MB
12	7	Accepted	0.00 sec, 1.65 MB
13	16	Accepted	0.00 sec, 1.66 MB

Question **5**

Correct

Mark 100.00 out of 100.00

Time limit

1 s

Memory limit

64 MB

List fibonacci adalah list yang elemennya merupakan jumlah dari 2 elemen sebelumnya. Pada awal program, program akan meminta jumlah elemen dari list fibonacci yang akan dibuat. Selanjutnya, program akan mengeluarkan output sesuai aturan berikut:

- Apabila jumlah elemen = 0, maka program akan langsung mengembalikan list kosong.
- Apabila jumlah elemen = 1, maka program akan meminta sebuah inputan yang menjadi elemen dalam list
- Apabila jumlah elemen  $\geq 2$ , maka program akan meminta dua buah inputan. Inputan pertama adalah elemen pertama dalam list dan inputan kedua adalah elemen kedua dalam list

Berikut adalah contoh input/output dari program

Input	Output
0	[]
1	[2]
2	[4]
1	[4]
4	[0,1]
2	[0,1]
0	[0,1]
1	[0,1]
4	[3,4,7,11]
3	[3,4,7,11]
4	[3,4,7,11]

Untuk mempermudah pengerjaan, silahkan lengkapi file [template.c](#) (boleh mengganti blok kode program yang telah ditulis pada template) dan submit dengan format nama listfibonacci.c

Note: Semua input berupa bilangan bulat positif

Diberikan ADT yang digunakan adalah:

- [listlinier.h](#)

- [listlinier.c](#)

C

 [listfibonacci.c](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	6	Accepted	0.00 sec, 1.59 MB
2	6	Accepted	0.00 sec, 1.71 MB
3	6	Accepted	0.00 sec, 1.62 MB
4	6	Accepted	0.00 sec, 1.60 MB
5	6	Accepted	0.00 sec, 1.59 MB

No	Score	Verdict	Description
6	8	Accepted	0.00 sec, 1.35 MB

7	6	Accepted	0.00 sec, 1.66 MB
8	6	Accepted	0.00 sec, 1.61 MB
9	6	Accepted	0.01 sec, 1.62 MB
10	6	Accepted	0.00 sec, 1.61 MB
11	6	Accepted	0.00 sec, 1.60 MB
12	6	Accepted	0.00 sec, 1.71 MB
13	6	Accepted	0.00 sec, 1.64 MB
14	6	Accepted	0.00 sec, 1.50 MB
15	16	Accepted	0.00 sec, 1.62 MB

Question **6**  
Correct  
Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Buatlah implementasi **removeDuplicate.c** dari file header removeDuplicate.h!  
Perhatian:

- List masukan sudah **terurut membesar**
- List hasil juga harus **terurut membesar**

- [removeDuplicate.h](#)
- [listlinier.h](#)
- [listlinier.c](#)
- [boolean.h](#)

C

 [removeDuplicate.c](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	20	Accepted	0.00 sec, 1.67 MB
2	20	Accepted	0.00 sec, 1.51 MB
3	20	Accepted	0.00 sec, 1.65 MB
4	20	Accepted	0.00 sec, 1.67 MB
5	20	Accepted	0.00 sec, 1.59 MB

