Started on	Thursday, 30 November 2023, 7:21 PM
State	Finished
Completed on	Thursday, 30 November 2023, 8:15 PM
Time taken	54 mins 5 secs
Grade	**96.00** out of 100.00

Question **1**

Correct

Mark 30.00 out of 30.00

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 dan cache sudah penuh, 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 memiliki ukuran maksimal N. Lalu akan ada Q buah operasi pengambilan nilai x:

- Jika cache kosong, nilai x dimasukkan ke cache.
- Jika x ada di cache, nilai x dipindah ke depan cache.
- Jika x tidak ada di cache dan cache belum penuh, nilai x dimasukkan ke depan cache.
- Jika x tidak ada di cache dan cache sudah penuh, nilai terakhir pada cache dihapus dan nilai x dimasukkan ke depan cache.

Untuk setiap operasi, tuliskan apakah operasi "hit" atau "miss". Lalu, tuliskan isi list. Lalu setelah seluruh operasi selesai dilakukan, tuliskan hit ratio di akhir. Hit ratio didapat dari jumlah hit dibagi jumlah hit dan jumlah miss. Tuliskan hanya dua angka di belakang koma.

Kumpulkan file **cache.c**

Note: Ukuran cache bisa nol.

Contoh masukan/keluaran

Masukan	**Keluaran**	**Keterangan**
3 8 4 6 7 5 5 1 5 7	miss [4] miss [6,4] miss [7,6,4] miss [5,7,6] hit [5,7,6] miss [1,5,7] hit [5,1,7] hit [7,5,1] hit ratio: 0.38	Pada awalnya, cache kosong []. Namun ukuran maksimal adalah 3. Ada 8 operasi yang dilakukan. Pada operasi pertama, 4 tidak ada di cache. Pada operasi kedua, 6 tidak ada di cache. Pada operasi ketiga, 7 tidak ada di cache. Pada operasi keempat, 5 tidak ada di cache, tetapi cache sudah penuh jadi nilai 4 dibuang. Pada operasi kelima, 5 ada di cache. Pada operasi keenam, 1 tidak ada di cache sehingga nilai 6 dibuang. Pada operasi ketujuh, 5 ada di cache sehingga nilai 5 dipindah ke depan. Pada operasi ketujuh, 7 ada di cache sehingga nilai 7 dipindah ke depan. Hit ratio: 3/(3+5) = 0.375 ≈ 0.38
5 0	hit ratio: 0.00	Tidak ada operasi yang dilakukan sehingga hit ratio = 0

C

cache.c

Score: 30

Blackbox

Score: 30

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	1.5	Accepted	0.00 sec, 1.64 MB
2	1.5	Accepted	0.00 sec, 1.55 MB
3	1.5	Accepted	0.00 sec, 1.65 MB
4	1.5	Accepted	0.00 sec, 1.71 MB
5	1.5	Accepted	0.00 sec, 1.55 MB
6	1.5	Accepted	0.00 sec, 1.73 MB
7	1.5	Accepted	0.00 sec, 1.71 MB
8	1.5	Accepted	0.00 sec, 1.55 MB
9	1.5	Accepted	0.00 sec, 1.68 MB
10	1.5	Accepted	0.00 sec, 1.70 MB
11	1.5	Accepted	0.00 sec, 1.68 MB
12	1.5	Accepted	0.00 sec, 1.50 MB
13	1.5	Accepted	0.00 sec, 1.65 MB
14	1.5	Accepted	0.00 sec, 1.55 MB
15	1.5	Accepted	0.00 sec, 1.64 MB
16	1.5	Accepted	0.00 sec, 1.70 MB
17	1.5	Accepted	0.00 sec, 1.66 MB
18	1.5	Accepted	0.00 sec, 1.64 MB
19	1.5	Accepted	0.00 sec, 1.73 MB
20	1.5	Accepted	0.00 sec, 1.68 MB

Question **2**

Partially correct

Mark 26.00 out of 30.00

Time limit	1 s
Memory limit	64 MB

Lengkapi ADT List Berkait dengan representasi fisik pointer yang telah dibuat sebagai tugas pra-praktikum dengan beberapa fungsi/prosedur yang diberikan dalam file **listlinier.h**, yaitu:

Pencarian sebuah elemen

1. boolean fSearch (List l, Address p);
2. Address searchPrec (List l, ElType x);

Pencarian Nilai Ekstrim dan Rata-Rata (prekondisi: list tidak kosong)

1. ElType maxValue (List l);
2. Address adrMax (List l);
3. ElType minValue (List l);
4. Address adrMin (List l);
5. float average (List l);

Proses terhadap semua elemen list

1. void deleteAll (List *l);
2. void copyList (List *l1, List *l2)
3. void inverseList (List *l);

Kumpulkan file **listlinier.c.**

C

listlinier.c

Score: 26

Blackbox

Score: 26

Verdict: Wrong answer

Evaluator: Exact

No	Score	Verdict	Description
1	2	Accepted	0.00 sec, 1.50 MB
2	2	Accepted	0.00 sec, 1.62 MB
3	2	Accepted	0.00 sec, 1.50 MB
4	2	Accepted	0.00 sec, 1.62 MB
5	2	Accepted	0.00 sec, 1.63 MB
6	2	Accepted	0.00 sec, 1.49 MB
7	0	Wrong answer	0.00 sec, 1.60 MB

No	Score	Verdict	Description
8	2	Accepted	0.00 sec, 1.60 MB
9	0	Wrong answer	0.00 sec, 1.66 MB
10	2	Accepted	0.00 sec, 1.71 MB
11	2	Accepted	0.00 sec, 1.61 MB
12	2	Accepted	0.00 sec, 1.47 MB
13	2	Accepted	0.00 sec, 1.50 MB
14	2	Accepted	0.00 sec, 1.60 MB
15	2	Accepted	0.00 sec, 1.55 MB

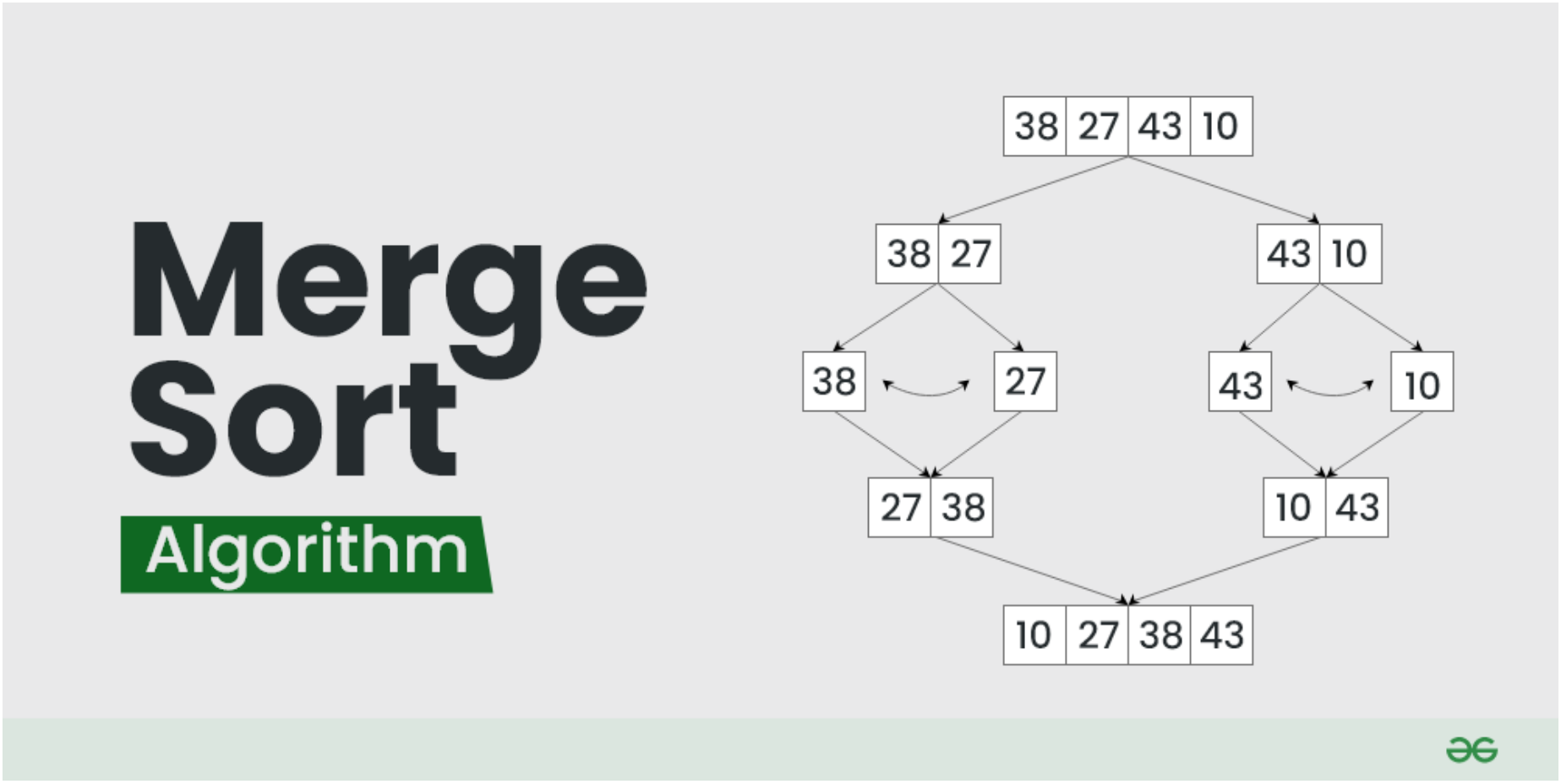
Question **3**

Correct

Mark 40.00 out of 40.00

Time limit	1 s
Memory limit	64 MB

Tuan Bus membaca sebuah artikel mengenai algoritma merge sort. Merge sort adalah algoritma sorting yang akan secara rekursif melakukan splitting sebuah list sampai tidak bisa dipecah lagi. Setelah itu, pecahan ini akan diurutkan menjadi sebuah sub-array, yang kemudian akan digabungkan menjadi satu kesatuan kembali. Dibawah ini disediakan gambar ilustrasi bagaimana merge sort bekerja.



Tuas Bus tertarik untuk mengimplementasikan algoritma ini menggunakan linked list. **Sorting yang dilakukan adalah ascending**. Tuan Bus sudah mempersiapkan file header **merge.h** berisi fungsi yang perlu diimplementasikan. Bantulah Tuan Bus untuk membuat program merge sort ini.

Kumpulkan file **merge.c**

Berikut contoh keluaran yang diharapkan

Masukan (pembacaan masukan tidak perlu dibuat)	Keluaran	Keterangan

Masukan (pembacaan masukan tidak perlu dibuat)	Keluaran	Keterangan
5 4 3 6 1 2	[4,3,6,1,2] [4,3,6] [4,3] [4] [3] [6] [1,2] [1] [2]	Pada rekursi pertama: List front = [4,3,6] List back = [1,2] Pada rekursi kedua, program berfokus ke list front [4,3,6]: List front baru = [4,3] List back = [6] Pada rekursi ketiga, program berfokus ke list front [4,3]: List front baru = [4] List back = [3] Karena 3 < 4, maka list baru mengandung [3] dan list back menjadi kosong. Karena list back kosong, maka kita tambahkan 4 ke dalam list, menjadi [3,4] Program kembali ke tahap rekursi kedua, dimana 3 < 6, maka list baru mengandung [3] dan list front berisi [4]. Setelah itu, program membandingkan 4 < 6, sehingga 4 ditambahkan ke dalam list, menjadi [3,4] dan list front menjadi kosong. Karena list front kosong, maka tambahkan 6 ke dalam list, menjadi [3,4,6] Proses ini akan kembali ke tahap rekursi pertama, dimana seluruh proses di atas akan terulang kembali, menghasilkan list [1,2,3,4,6]

C

merge.c

Score: 40

Blackbox

Score: 40

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	8	Accepted	0.00 sec, 1.64 MB
2	8	Accepted	0.00 sec, 1.56 MB
3	8	Accepted	0.00 sec, 1.62 MB
4	8	Accepted	0.00 sec, 1.64 MB
5	8	Accepted	0.00 sec, 1.66 MB

◄ ADT Linked List - Praktikum

Jump to...

Pra Praktikum 9 ►