Started on	Thursday, 13 October 2022, 6:12 PM
State	Finished
Completed on	Thursday, 13 October 2022, 7:12 PM
Time taken	59 mins 21 secs
Marks	350.00/540.00
Grade	**64.81** out of 100.00

Question **1**

Correct

Mark 240.00 out of 240.00

Time limit	1 s
Memory limit	64 MB

Kumpulkan **queue.c**

C

queue.c

Score: 240

Blackbox

Score: 240

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.00 sec, 1.54 MB
2	10	Accepted	0.00 sec, 1.42 MB
3	10	Accepted	0.00 sec, 1.51 MB
4	10	Accepted	0.00 sec, 1.54 MB
5	10	Accepted	0.00 sec, 1.50 MB
6	10	Accepted	0.00 sec, 1.54 MB
7	10	Accepted	0.00 sec, 1.54 MB
8	10	Accepted	0.00 sec, 1.49 MB
9	10	Accepted	0.00 sec, 1.54 MB
10	10	Accepted	0.00 sec, 1.50 MB
11	10	Accepted	0.00 sec, 1.49 MB
12	10	Accepted	0.00 sec, 1.50 MB
13	10	Accepted	0.00 sec, 1.46 MB
14	10	Accepted	0.00 sec, 1.57 MB
15	10	Accepted	0.00 sec, 1.50 MB
16	10	Accepted	0.00 sec, 1.51 MB
17	10	Accepted	0.00 sec, 1.46 MB
18	10	Accepted	0.00 sec, 1.51 MB
19	10	Accepted	0.00 sec, 1.42 MB
20	10	Accepted	0.00 sec, 1.54 MB
21	10	Accepted	0.00 sec, 1.49 MB
22	10	Accepted	0.00 sec, 1.47 MB
23	10	Accepted	0.00 sec, 1.51 MB

No	Score	Verdict	Description
24	10	Accepted	0.00 sec, 1.49 MB

Question **2**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Kumpulkan file queue.c dan antrianSix.c dikumpulkan sebagai antrianSix.zip

Ambillah ADT Queue yang direpresentasikan menggunakan array eksplisit dan alokasi memori statik yang telah Anda kerjakan sebagai tugas pra-praktikum. Kapasitas dari queue sesuai header (100).

Widya adalah penggemar mata kuliah Algoritma dan Struktur Data. Suatu hari, Widya ingin mengeksplorasi lebih mengenai implementasi struktur data queue pada dunia nyata. Oleh karena itu, Widya memutuskan untuk membuat simulasi sederhana antrian PRS pada SIX. Buatlah sebuah program utama yang menggunakan ADT Queue yang telah dibuat untuk membantu Widya mensimulasikan antrian PRS. Elemen queue menyatakan lama waktu mahasiswa mengantri di sistem. Program harus melakukan hal sebagai berikut:

1. Program dimulai dengan loop, pengguna bisa memilih untuk menambah mahasiswa dalam antrian atau menghapus mahasiswa (menghapus mahasiswa berarti mahasiswa tersebut dilayani). Pengguna memiliki opsi sebagai berikut:
    1. Jika memilih menambah antrian, masukkan kode operasi = 1 diikuti panjang waktu mengantri (elemen queue). Contoh: 1 5 (berarti menambah elemen queue dengan waktu antrian = 5 menit).
    2. Untuk melayani seorang mahasiswa, dilakukan penghapusan elemen antrian dengan memasukkan kode operasi = 2. Penambahan dan penghapusan harus mengikuti ketentuan operasi enqueue dan dequeue. Perhatikan untuk kasus queue kosong dan queue penuh. Pada operasi enqueue, jika queue penuh, tidak terjadi penambahan elemen dan berikan pesan "Queue penuh". Pada operasi dequeue, jika queue kosong, tidak terjadi penghapusan elemen dan berikan pesan "Queue kosong".
    3. Untuk menghentikan proses, masukkan kode operasi = 0.
2. Di akhir program, ditampilkan statistik mahasiswa yang telah dilayani, yaitu jumlah mahasiswa yang telah dilayani, waktu minimal mengantri, waktu maksimal mengantri, dan waktu rata-rata mengantri (dalam format dua angka di belakang koma). Apabila belum ada mahasiswa yang telah dilayani, waktu minimal, maksimal, dan rata-rata memberikan pesan "Tidak bisa dihitung".

Petunjuk: Untuk mengeluarkan float dengan 2 angka di belakang koma, gunakan **printf("%.2f", var).**

Berikut adalah contoh interaksi input/output program (Garis bawah adalah input):

Input/Output	**Keterangan**
1 5 1 3 0 0 Tidak bisa dihitung Tidak bisa dihitung Tidak bisa dihitung	 Jumlah mahasiswa yang telah dilayani = 0 Waktu minimal, maksimal, dan rata-rata tidak ada karena tidak ada mahasiswa yang dilayani
1 5 1 3 2 0 1 5 5 5.00	 Jumlah mahasiswa yang telah dilayani = 1 Waktu minimal = 5 Waktu maksimal = 5 Waktu rata-rata = 5

1 5 1 3 2 2 2 Queue kosong 0 2 3 5 4.00	Jumlah mahasiswa yang telah dilayani = 2 Waktu minimal = 3 Waktu maksimal = 5 Waktu rata-rata = (5+3)/2 = 4.00
... 1 10 1 10 1 10 1 10 Queue penuh 1 9 Queue penuh 2 2 1 5 ...	Kasus Queue Penuh

C

antrianSix.zip

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	12.5	Accepted	0.00 sec, 1.66 MB
2	12.5	Accepted	0.00 sec, 1.60 MB
3	12.5	Accepted	0.00 sec, 1.71 MB
4	12.5	Accepted	0.00 sec, 1.71 MB
5	12.5	Accepted	0.00 sec, 1.64 MB
6	12.5	Accepted	0.00 sec, 1.71 MB
7	12.5	Accepted	0.02 sec, 1.55 MB
8	12.5	Accepted	0.00 sec, 1.70 MB

Question **3**

Partially correct

Mark 10.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Kumpulkan file prioqueuetime.c

Priority queue adalah struktur antrian dengan prioritas sehingga elemen-elemen queue disusun terurut berdasarkan prioritas tersebut. Pada kasus ini, setiap elemen queue terdiri dari dua komponen:

```
type infotype : <
```

```
time : integer[1..100], { waktu dengan nilai 1..100 (1 adalah waktu
terendah) }
```

```
name : char { elemen karakter }
```

```
>
```

dan setiap elemen queue terurut membesar berdasarkan time, sehingga elemen dengan time lebih rendah diantrikan di depan elemen dengan time lebih tinggi.

Diberikan header ADT PrioQueueTime dalam file prioqueuetime.h. Buatlah implementasi ADT PrioQueueTime (file: prioqueuetime.c) yang diadaptasikan dari ADT Queue yang telah Anda kerjakan sebagai tugas pra-praktikum.

Berikut beberapa perubahan berikut terhadap ADT PrioQueueTime:

- **Elemen queue (type infotype) dan selektor.**
- **Prosedur Add:**

Proses penambahan elemen dilakukan sesuai dengan time. Elemen baru akan ditambahkan di depan seluruh elemen queue yang memiliki time yang lebih tinggi dan di belakang seluruh elemen queue yang memiliki time lebih rendah. Jika time sama, maka elemen baru akan ditambahkan sebagai elemen yang terakhir pada prioritas tersebut.

Contoh: PQ = [<1,'j'>, <2,'d'>, <2,'g'>, <4,'o'>]

Add(PQ,<2,'a'>) ? PQ = [<1,'j'>, <2,'d'>, <2,'g'>, <2,'a'>, <4,'o'>]

Add(PQ,<4,'y'>) PQ = [<1,'y'>, <2,'d'>, <2,'g'>, <4,'o'>, <4,'y'>]

- **Prosedur PrintQueue:**

Berfungsi mencetak ke layar seluruh elemen queue, satu elemen per baris, time dan name dipisah dengan satu spasi. Tambahkan tanda # (dan enter) di baris terakhir. Jika queue kosong, pada layar hanya dicetak tanda #.

Contoh:

Isi PQ	**PrintQueue (PQ)**
[<1,'j'>,<2,'d'>,<2,'g'>,<4,'o'>]	1 j 2 d 2 g 4 o #
[]	#

C

prioqueuetime.c

Score: 10

Blackbox

Score: 10

Verdict: Wrong answer

Evaluator: Exact

No	Score	Verdict	Description
1	0	Wrong answer	0.00 sec, 1.65 MB
2	10	Accepted	0.00 sec, 1.54 MB
3	0	Wrong answer	0.00 sec, 1.63 MB
4	0	Wrong answer	0.00 sec, 1.63 MB
5	0	Wrong answer	0.00 sec, 1.54 MB
6	0	Wrong answer	0.00 sec, 1.64 MB
7	0	Wrong answer	0.00 sec, 1.60 MB
8	0	Wrong answer	0.00 sec, 1.61 MB
9	0	Wrong answer	0.00 sec, 1.64 MB
10	0	Wrong answer	0.00 sec, 1.66 MB

Question **4**

Not answered

Marked out of 100.00

Time limit	1 s
Memory limit	64 MB

Kumpulkan file **prioqueuetime.c** dan **sjf.c** dikumpulkan sebagai **sjf.zip**

Pada setiap PC, OS menggunakan suatu algoritma CPU *scheduling* untuk menentukan proses mana yang akan menggunakan CPU untuk eksekusi. Kegunaan utama dari CPU *scheduling* ini adalah memastikan setiap kali CPU *idle*, OS dapat memilih salah satu proses dari queue untuk dieksekusi.

Dengan memanfaatkan ADT PrioQueueTime yang telah anda buat pada soal sebelumnya, buatlah sebuah program yang mensimulasikan CPU *scheduling* pada suatu OS menggunakan algoritma SJF (Shortest Job First). Pada algoritma ini, proses dengan *execution time* terpendek akan dipilih terlebih dulu.

Pada program ini, program membaca N, banyaknya proses yang akan dilakukan. Selanjutnya terdapat N baris dengan baris ke-i menandakan proses yang diterima pada detik ke-i. Proses ditulis dengan format <waktu_eksekusi> <nama_proses>.

Program Anda akan menuliskan proses yang mulai dijalankan beserta dengan detik proses mulai. Format penulisan adalah <detik_mulai> <nama_proses>. Apabila ada sebuah proses sedang berjalan, maka proses lain menunggu di queue dan program tidak mengeluarkan output apa-apa. Apabila proses terakhir sudah di-*input*, program akan menuliskan sisa proses yang belum dijalankan dengan format detik berapa ia akan mulai dan nama prosesnya. Sebagai contoh, perhatikan tabel sebagai berikut:

Input	**Output**	**Keterangan**
6 2 o 2 n 1 c 4 i 2 d 3 g	0 o 2 c 3 n 5 d 7 g 10 i	Terdapat 6 proses
		Dijalankan proses o pada detik ke - 0.
		Memasukkan proses n pada queue di detik ke - 1.
		Proses o selesai, proses c dimasukkan pada queue dan dijalankan pada detik ke-2.
		Proses i dimasukkan pada queue dan proses n dijalankan pada detik ke - 3.
		Proses d dimasukkan pada queue pada detik ke - 4.
		Proses g dimasukkan pada queue pada detik - 5, dan menuliskan sisa proses dan detik dia akan mulai.

C