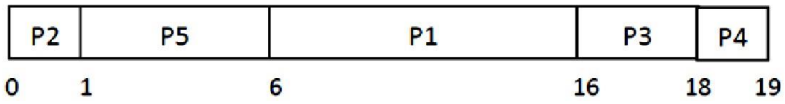
Sheet #4 (Linked Queue)

1. In some systems, a priority is associated with each process and the CPU is allocated to the process with the highest priority (smallest value). Equal priority processes are scheduled in FIFO order.

Ex:	Process	Burst Time	Priority
	P1	10	3
	P2	1	1
	P3	2	3
	P4	1	4
	P5	5	2

They will be scheduled to CPU as follows:



Define a suitable data structure, and then write a simulation program for the system described above. The program should display the following menu:

1. Add a New Process.
2. Serve a Process.
3. Number of Waiting Processes.
4. Exit menu.

Hints:

- You have to adjust the **LINKED QUEUE ADT** in implementation level to be suitable for solving this problem.
- You have to track the number of waiting processes "as integer **size**".
- The process should have the following fields: process ID and priority.

Sheet #6 (Linked Lists Use Cases)

Use Case 1 (A Mobile Telephone Service Office)

One of the main elements in the cellular mobile systems is the Mobile Telephone Service Office (MTSO). It provides interfacing of the mobile system to the public switched telephone network. Many users can use their cellular phones simultaneously, and the MTSO must handle these competing calls. The FIFO policy is not always fair, since more important calls must be served promptly. For example, if there are some voice calls currently in the system –either waiting for connection or already connected- any emergency call must have a priority over them. In addition, any voice call must have a priority over data calls.

Since this sheet is educational at a very introductory level, we will consider an over-simplified simulation for the prioritization system mentioned above. Assume that:

- The system handles only three types of calls: *Data Calls, Voice Calls, and Emergency Calls*. Each call has a sender, and a receiver IDs.
- Any user starts using his cellular phone and placing a voice call, he should have the least priority in the system with respect to other voice and emergency calls. However, it is fair enough to have higher priority among other data calls. This means that the new voice call should be inserted in the data structure lagging all the existing voice and emergency calls but leading all the existing data calls.
- Similarly, a new emergency call should be inserted in the data structure lagging all the already-existing emergency calls but leading all other calls.
- A new data call is always inserted in the structure at the rear.
- A new call is placed/dismissed by selecting an option in the menu of the main program.

Write a simulation program for the system described above. The program should display the following menu.

1. Data Call coming.
2. Voice Call coming.
3. Emergency Call coming.
4. Serve a call.
5. Dismiss calls with least priority.
6. Exit.

Hints:

Each call should have the following fields: a tag (D, V, or E), the number of the calling party, and the number of the party to be called. In addition, a Data call has an extra one field (number of packets); and a voice call has an extra two fields (a Boolean variable for showing the caller ID and another Boolean variable for roaming option). These fields should be entered for every call.

The option "Serve a call" should display the call at the front of the queue then delete it from the system.

The option "Dismiss all of the least prior calls" should delete from the system all the data calls, if any in the system; otherwise, it should dismiss all the voice calls.

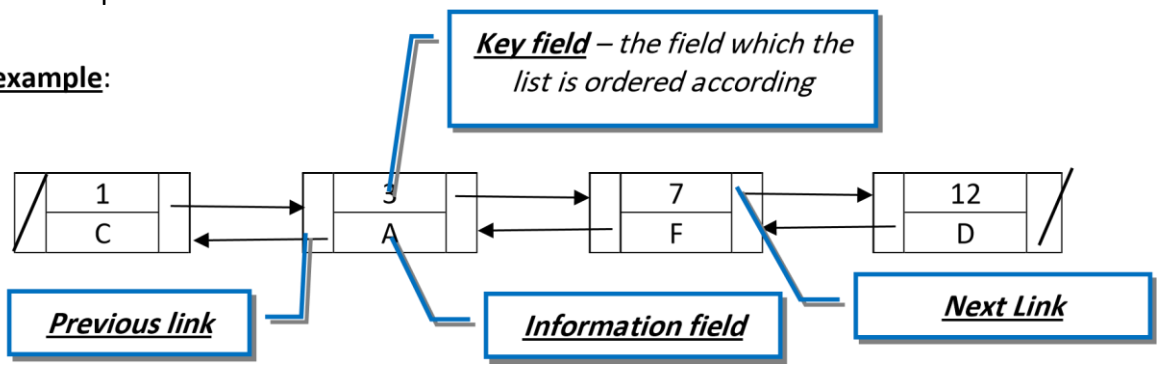
"Exit" should save the contents of the data structure into a file then quits. In addition, when the program starts it should create a new structure and load it from the same file. This requirement has no practical value; it only helps in saving the simulation information for the next run.

Linked List Types

A **doubly sorted linked list** is a linked data structure that consists of a set of sequentially linked nodes. Data in the list should be sorted according to a specific key. Each node contains:

- a data entry field that contain a key and information,
- and two pointers that are references to the previous and to the next node in the sequence of nodes.

For example:



You are required to:

1. Write the type definition of a doubly sorted linked list node in C.
2. Define the doubly linked list to be, simply, a pointer to a node.
3. Write a C function to initialize the list.
4. Write a C function to insert an element.

5. Write a C function to delete an element of a specific key after retrieving its information.
6. Write a C function to destroy the list.

Remember to use appropriate names for types and variables. It is expected that you define the pre-conditions and post-conditions for every function of the above.