CS201 Spring 2019
Assignment #3: Priority Queue
Due 11:59 pm Monday, Feb. 25th
15 points

A priority queue is a FIFO list, maintained in sorted priority order. For this assignment, keep the list in ascending priority order (so that nodes having lower priority go in the front of the queue).

Put your code in pqueue.netid.c and pqueue.netid.h

Define these structs:

```
typedef struct {
  int id;
  char name[32];
} StudentRecord;

typedef struct PQueueStruct {
  int priority;
  void *data;
  struct PQueueStruct *next;
} PQueueNode;
```

Implement the following functions

```
int enqueue(PQueueNode **pqueue, int priority, void *data)
void *dequeue(PQueueNode **pqueue)
void *peek(PQueueNode *pqueue)
void printQueue(PQueueNode *pqueue, void (printFunction)(void*))
int getMinPriority(PQueueNode *pqueue)
int queueLength(PQueueNode *pqueue)
void printStudentRecord(void *data)
```

`(1) enqueue(PQueueNode **pqueue, int priority, void *data)`

This will put the data in the priority queue in the correct place. If there is already one or more nodes in the list having the same priority as the data that you are enqueueing, then put the new node after all of the nodes having that priority. Return zero from this function.

`(2) void *dequeue(PQueueNode **pqueue)`

Remove the front of the list and return the data from that list node. If the pqueue is empty, then return NULL.

`(3) void *peek(PQueueNode *pqueue)`

Return the data from the first node in the pqueue. If the pqueue is empty, then return NULL. The peek operation does not actually remove a node from the pqueue!

```
(4) void printQueue(PQueueNode *pqueue, void (printFunction)(void*))
```

Print each the data from each node in the queue.

```
(5) int getMinPriority(PQueueNode *pqueue)
```

Return the priority of the first node in the pqueue. If the pqueue is empty, return -1.

```
(6) int queueLength(PQueueNode *pqueue)
```

Return the number of nodes in the pqueue.

```
(7) void printStudentRecord(void *data)
```

Print an instance of StudentRecord as "%s %d", with the name field and id field.

Passing a function as a parameter

For an example of how to use a function as a parameter, see function-parameter.c on the course gitlab site.

Testing your code

Here is are the operations you should do to test this:

create a student named "Chris Paul", with id = 1245
enqueue this student, with priority = 5
create a student named "James Harden", with id = 2324
enqueue this student, with priority = 3
create a student named "Kevin Love", with id = 3794
enqueue this student, with priority = 8
create a student named "Dwyane Wade", with id = 8345
enqueue this student, with priority = 7
create a student named "Luol Deng", with id = 3245
enqueue this student, with priority = 1
create a student named "Al Horford", with id = 4326
enqueue this student, with priority = 1
create a student named "Jayson Tatum", with id = 1287
enqueue this student, with priority = 8

Then print your queue. You should see this:

```
priority = 1 data = Luol Deng 3245
priority = 1 data = Al Horford 4326
priority = 3 data = James Harden 2324
priority = 5 data = Chris Paul 1245
priority = 7 data = Dwyane Wade 8345
priority = 8 data = Kevin Love 3794
priority = 8 data = Jayson Tatum 1287
```

Now, do the following:

peek: should get Luol Deng 3245
dequeue from the queue and print the student record: it should be Luol Deng 3245
peek: should get Al Horford 4326
dequeue from the queue and print the student record: it should be Al Horford 2324
peek: the result should be James Harden 2324

create a student named "Donovan Mitchell", with id = 9334
enqueue this student, with priority = 2
create a student named "Kyle Korver", with id = 7328
enqueue this student with priority = 3

Then print the queue. You should see this:

```
priority = 2 data = Donovan Mitchell 934
priority = 3 data = James Harden 2324
priority = 3 data = Kyle Korver 7328
priority = 5 data = Chris Paul 1245
priority = 7 data = Dwyane Wade 8345
priority = 8 data = Kevin Love 3794
priority = 8 data = Jayson Tatum 1287
```

Finally, do this:

```
// studentRec is a variable of type StudentRecord*
// minPriority is a variable of type int

while (queueLength(pqueue) > 0) {
  minPriority = getMinPriority(pqueue);
  printf("min priority = %d\n", minPriority);
  studentRec = dequeue(&pqueue);
  printf("dequeued: ");
  printStudentRecord(studentRec);
}
```

and you should see this output:

```
min priority = 2
dequeued: Donovan Mitchell 934
min priority = 3
dequeued: James Harden 2324
min priority = 3
dequeued: Kyle Korver 7328
min priority = 5
dequeued: Chris Paul 1245
```

```
min priority = 7
dequeued: Dwyane Wade 8345
min priority = 8
dequeued: Kevin Love 3794
min priority = 8
dequeued: Jayson Tatum 1287
```