Adding the following 10 tasks, all with priority of 1 in the following order:

1. Cake
2. Pop
3. Cookie
4. Mushroom
5. Apple
6. Gum
7. Candy
8. Yam
9. Watermelon
10. Jello

Results of printing the list (same order for removal since printing references remove min):

- 1, cake
- 1, jello
- 1, watermelon
- 1, yam
- 1, candy
- 1, gum
- 1, apple
- 1, mushroom
- 1, cookie
- 1, pop

This is not what I expected, since it's a partial reversal of the list. This order makes sense when you remember that this is a heap though. Since there's no other way to order the list, the tree fills in from left to right, so there's no way to preserve order.

For me, I would expect items of the same priority to be added and removed in a FIFO ordering. To achieve this, I implemented a time stamp system. This gives a second layer of comparison for items with the same priority. The timestamp starts at 0 and increments each time the menu/switch function runs. The current time stamp is passed to the newTask function and assigned to that new task. Thus, even if two tasks have the same priority, it should be impossible for them to have the same priority. When a user loads a list, a unique timestamp is given to each item as it's loaded from the file.

I also adjusted the taskCompare function. If the compare function finds two equivalent priorities, the function then compares the timestamps, returning values parallel to comparing the priorities.

Together, all of this created a FIFO operation for the priority heap. I added the values in the first list and removed them. Both actions maintained the order from the first numbered list above.