```
#include <iostream>
#include<fstream>
#include<string>

using namespace std;

struct RecType
{
   int priority;
   string name;
};

///Using a class
class pque
{
public:
   ///1st group of public variables used for the functions in the private
   RecType x[100];
   int lastIndex;
   RecType userInput;

private:
   ///modified and updated version of the record heap array
   void maxreheapifyUpward(RecType x[], int lastIndex)
   {
      int parentIndex;
      int childIndex = lastIndex;
      while (childIndex > 0)
      {
         parentIndex = childIndex / 2;
         if (x[childIndex].priority <= x[parentIndex].priority)
```

```
        break;
        else
        {
           ///swap values at child and at parent.
           swap(x, parentIndex, childIndex);
           ///Update child to parent
           childIndex = parentIndex;
        }
     }
  }

  ///same as the above
  void maxreheapifyDownward(RecType x[], int lastIndex)
  {
     int parentIndex = 0;
     int largeChildIndex;
     ///cout << "hi maxreheapifyDownward" << endl;
     ///while the parent index is still less than the last index
     while (parentIndex < lastIndex)
     {
        ///find the largest index
        largeChildIndex = findLargeChildIndex(parentIndex, lastIndex);
        if (largeChildIndex < 0 || x[largeChildIndex].priority <= x[parentIndex].priority)
        break;
        else
        {
           ///swap value at parentIndex with value at largeChildIndex
           swap(x, parentIndex, largeChildIndex);
           ///and update the parentIndex
           parentIndex = largeChildIndex;
        }
```

```
    }
  }

  int findLargeChildIndex(int parentIndex, int lastIndex)
  {
    int lChildIndex = (2 * parentIndex) + 1;
    int rChildIndex = (2 * parentIndex) + 2;

    ///if both children exist
    if (rChildIndex <= lastIndex && lChildIndex <= lastIndex)
    {
      ///compare the values of the right and left child
      ///and return the index where the value is smaller
      if (x[rChildIndex].priority>x[lChildIndex].priority)
        return rChildIndex;
      else
        return lChildIndex;
    }
    ///if only the left child exists
    else if (lChildIndex <= lastIndex)
    {
      return lChildIndex;
    }
    ///when none of them exists
    else
    {
      return -1;
    }
  }

public:
```

```
///2nd half of the public variables and functions
///seperated as it needs the first 2 variables in the first public and private
///to bee declared first
pque()
{
   ///set the last index to -1 (end)
   lastIndex = -1;
}

///enque function
void penque (int p, string n)
{
   lastIndex++;
   x[lastIndex].priority = p;
   x[lastIndex].name = n;
   maxreheapifyUpward(x,lastIndex);
}


///deque function
RecType pdeque()
{
   RecType returnValue = x[0];
   x[0] = x[lastIndex];
   lastIndex--;
   maxreheapifyDownward(x, lastIndex);
   return returnValue;
}

///swap function
void swap(RecType x[], int parentIndex, int childIndex)
```

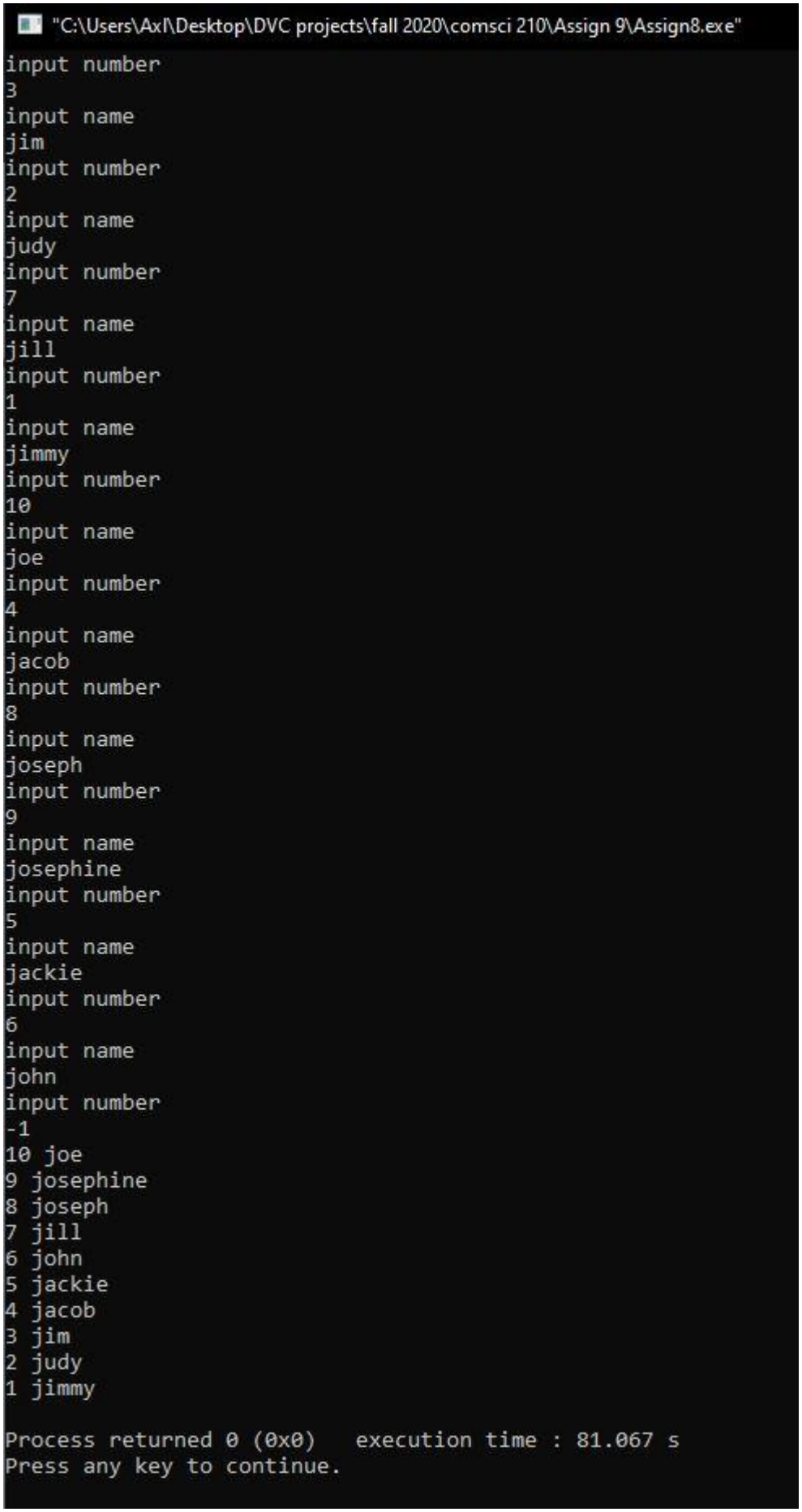
```
    {
        RecType t;
        t = x[parentIndex];
        x[parentIndex] = x[childIndex];
        x[childIndex] = t;
    }

    ///function to check if it is empty
    bool isEmpty()
    {
        if (lastIndex < 0)
        return true;
        else
        return false;
    }
};

///driver funnction from the assignment
int main()
{
    pque recordList;
    while (recordList.userInput.priority >= 0)
    {
        cout << "input number" << endl;
        cin >> recordList.userInput.priority;
        if (recordList.userInput.priority == -1)
        {
            break;
        }
        cout << "input name" << endl;
        cin >> recordList.userInput.name;
```

```
        recordList.penque(recordList.userInput.priority, recordList.userInput.name);
    }


    while (!recordList.isEmpty())
    {
      recordList.userInput = recordList.pdeque();
      cout << recordList.userInput.priority << " " << recordList.userInput.name << endl;
    }
    return 0;
}
```


"C:\Users\Axl\Desktop\DVC projects\fall 2020\comsci 210\Assign 9\Assign8.exe"
input number
3
input name
jim
input number
2
input name
judy
input number
7
input name
jill
input number
1
input name
jimmy
input number
10
input name
joe
input number
4
input name
jacob
input number
8
input name
joseph
input number
9
input name
josephine
input number
5
input name
jackie
input number
6
input name
john
input number
-1
10 joe
9 josephine
8 joseph
7 jill
6 john
5 jackie
4 jacob
3 jim
2 judy
1 jimmy
Process returned 0 (0x0) execution time : 81.067 s
Press any key to continue.