**CS 302 – Assignment #8**

Purpose: Learn to about priority queues.
Due: Tuesday (3/21)
Points: 125 Part A → 75 pts, Part B → 50 pts

**Assignment - Part A:**


In mathematics, the $n^{th}$ Taxicab number[1] is defined as the smallest number that can be expressed as a sum of two positive cubes numbers in ***n*** distinct ways. The most famous taxicab number is 1729 = Ta(2) = $1^3 + 12^3 = 9^3 + 10^3$. For our purposes, we will use a less strict definition → a number that can be expressed as a sum of two positive cubes numbers in at least *two* distinct ways.

One algorithm for finding taxicab numbers uses a priority queue. As such, you should develop a C++ class to implement a priority queue.

Once the priority queue is developed and tested, create a main program to use the priority queue to find the integers that can be expressed as the sum of cubes. Write a main program that reads in a command line parameter, ***limit***, and prints out all nontrivial solutions of $a^3 + b^3 = c^3 + d^3$ such that ***a***, ***b***, ***c***, and ***d***, are less than or equal to ***limit***.


The ***limit*** should be read from the command line and must be verified to be between 10 and 1,000,000 (inclusive). If the limit specifier or limit value are invalid, an appropriate error message should be displayed. Refer to the example executions for formatting.

**Part B:**


When completed, create and submit a write-up (PDF format) not too exceed ~750 words including the following:

- Name, Assignment, Section
- Summary of the implemented priority queue data structure.
- Big-Oh for the priority queue operations (insert, deleteMin, reheapUp, reheapDown, buildHeap).
- Explain the difference between repeated inserting and using buildHeap.

1 For more information, refer to: https://en.wikipedia.org/wiki/Taxicab_number

**Submission:**

- Submit a compressed zip file of the program source files, header files, and makefile via the on-line submission by 23:50.

All necessary files must be included in the ZIP file. The grader will download, uncompress, and type **make** (so you must have a valid, working *makefile*).

**Relaxed Taxicab Number Algorithm**

Algorithm to find all nontrivial solutions of $a^3 + b^3 = c^3 + d^3$ such that ***a***, ***b***, ***c***, and ***d***, are less than or equal to ***limit***. Assuming the use of a structure, ***cubeSet***, containing ***a***, ***b***, and ***cubeSum***.

```
for i = 1 .. limit
        thisCubeSet = i, i,  i^3 + i^3
        priority =  i^3 + i^3
        pq.insert(thisCubeSet, priority)

count = 1
prevCubeSet = (0, 0, 0)
while not pq.empty()
        pq.deleteMin(currCubeSet, priority)
        if  currCubeSet.cubeSum == prevCubeSet.cubeSum
                count++
                if (count == 2)
                        display Taxicab number that can be expressed as
                                currCubeSet.a^3 + currCubeSet.a^3  and
                                        prevCubeSet.a^3 + prevCubeSet.b^3
        else
                count = 1
        prevCubeSet = curr CubeSet
        if  currCubeSet.b < limit
                priority =  currCubeSet.a^3 + currCubeSet.b+1^3
                newCubeSet = (currCubeSet.a, currCubeSet.b + 1, priority)
                pq.insert(newCubeSet, priority)
```

**Class Descriptions**

- Priority Queue Class
  The priority queue template class will implement functions specified below.

| **priorityQueue<myType>** |
|---|
| -heapNode: struct |
| -priority: unsigned long long |
| -item: myType |
| -count: int |
| -heapSize: int |
| -*myHeap: heapNode |

| |
|---|
| +priorityQueue(int=5000) |
| +~priorityQueue() |
| +entries() const: int |
| +insert(const myType, const unsigned long long): void |
| +deleteMin(myType &, unsigned long long &): bool |
| +isEmpty() const: bool |
| +printHeap() const: void |
| +readData(const string): bool |
| -reheapUp(int): void |
| -reheapDown(int): void |
| -buildHeap(): void |
| -resize(): void |

**Function Descriptions**

- The *priorityQueue()* constructor should initialize the binary heap to an empty state. The parameter must be checked to ensure it is at least 5000 and, if not, use the default value.
- The *~priorityQueue()* destructor should delete the heap.
- The *entries()* function should return the total count of elements in the heap.
- The *insert()* function should insert an entry into the binary heap. If the count of heap entries exceeds the heap size, the heap must be expanded via the private *resize()* function. The heap properties must be maintained via the private *reheapUp()* function. The count should be updated.
- The private *buildHeap()* function should update the heap to apply the heap properties.
- The *isEmpty()* function should return true if there are no elements in the heap and false otherwise.
- The *deleteMin()* function should remove the minimum entry from the heap. The heap properties must be maintained via the private *reheapDown()* function. Additionally, the count should be updated. If the heap is already empty, the function should return false and otherwise return the minimum priority information (via reference) and return true.
- The *printHeap()* function should print the current heap in level order with a blank line between each level. Refer to the sample output for an example of the formatting.
- The *readData()* function should read a formatted file (name and priority) and enter the values read into the array **without** using the *insert()* function. The size will need to be checked and the *resize()* function called as needed. Once the values are read, the *buildHeap()* function should be called. The trade-offs associated with this process should be noted in the write-up.
- The *reheapUp()* function to recursively ensure the heap order property is maintained. Starts at tree leaf and works up to the root. Must be written recursively.
- The *reheapDown()* function to recursively ensure the heap order property is maintained. Starts at the passed node and works down to the applicable leaf. Must be written recursively.
- The *resize()* function should create a new heap array twice the size of the existing heap, copy all entries from the current heap into the new heap, and delete the old heap. The *heapSize* should be updated accordingly.

Refer to the example executions for output formatting. Make sure your program includes the appropriate documentation. See Program Evaluation Criteria for CS 302 for additional information. ***Note, points will be deducted for especially poor style or inefficient coding.***

**Test Script**
A test script which will be used for scoring the final submission is provided for reference.

**Example Execution – Testing Program:**
Below is an example program execution for the testing program.

```
ed-vm% ./pqTest
****************************************************************
CS 302 - Assignment #98
Priority Queue Tester


================================================================
Test Set #0
----------------------------------------
PQ Heap (level order):
google  1

amazon  2
newegg  3

apple  4
dell  5
oracle  6
cisco  7

jupiter  8
belkin  9
ebay  10

----------------------------------------
PQ Heap Size: 10
Priority Order:
google  1
amazon  2
newegg  3
apple  4
dell  5
oracle  6
cisco  7
jupiter  8
belkin  9
ebay  10

================================================================
Test Set #1
----------------------------------------
PQ Heap (level order):
g  1

x  1
y  1

o  2
n  5
p  7
d  9

a  3
c  4
j  6
b  8
e  11
```

```
f  12
h  13
k  14

m  15

---------------------------------------
PQ Heap Size: 16
Priority Order:
g  1
x  1
y  1
o  2
a  3
c  4
n  5
j  6
p  7
b  8
d  9
e  11
f  12
h  13
k  14
m  15

================================================================
Test Set #3
Large test successful.

================================================================
Test Set #4
Email (1):  leo@MorbimetusVivamus.com
Email (2):  non.cursus@sapien.edu
Email (3):  ut.ipsum.ac@et.org
Email (4):  nec.ante@eteuismodet.edu
Email (5):  netus@nonummy.com
Email (6):  sit.amet.risus@tacitisociosquad.net
Email (7):  nascetur.ridiculus.mus@DonecestNunc.com
Email (8):  sed.dolor.Fusce@ornaretortorat.org
Email (9):  id.risus@non.co.uk
Email (10):  urna.justo@lectus.ca

****************************************************************
Game Over, thank you for playing.
ed-vm%
```

**Example Execution:**

Below is an example program execution for the main program.
*Note*, the **ed-vm%** is prompt on my machine. Your prompt will be different.

```
ed-vm%./cubeSum
Usage: ./cubeSum -l <limit>
ed-vm%
ed-vm% ./cubeSum -l 4
Error, invalid limit value.
ed-vm%
ed-vm%  ./cubeSum -l 2000000
Error, invalid limit value.
ed-vm%
ed-vm% ./cubeSum -x 200
Error, invalid limit specifier.
ed-vm%
```

```
ed-vm%
ed-vm% ./cubeSum -l 150
**********************************************************
CS 302 - Assignment #8
Relaxed Taxicab Numbers Program.

cube sum= 1729 =   9^3 + 10^3 = 1^3 + 12^3  cnt= 2
cube sum= 4104 =   9^3 + 15^3 = 2^3 + 16^3  cnt= 2
cube sum= 13832 =   2^3 + 24^3 = 18^3 + 20^3  cnt= 2
cube sum= 20683 =   10^3 + 27^3 = 19^3 + 24^3  cnt= 2
cube sum= 32832 =   4^3 + 32^3 = 18^3 + 30^3  cnt= 2
cube sum= 39312 =   2^3 + 34^3 = 15^3 + 33^3  cnt= 2
cube sum= 40033 =   16^3 + 33^3 = 9^3 + 34^3  cnt= 2
cube sum= 46683 =   3^3 + 36^3 = 27^3 + 30^3  cnt= 2
cube sum= 64232 =   17^3 + 39^3 = 26^3 + 36^3  cnt= 2
cube sum= 65728 =   31^3 + 33^3 = 12^3 + 40^3  cnt= 2
cube sum= 110656 =   4^3 + 48^3 = 36^3 + 40^3  cnt= 2
cube sum= 110808 =   6^3 + 48^3 = 27^3 + 45^3  cnt= 2
cube sum= 134379 =   12^3 + 51^3 = 38^3 + 43^3  cnt= 2
cube sum= 149389 =   29^3 + 50^3 = 8^3 + 53^3  cnt= 2
cube sum= 165464 =   20^3 + 54^3 = 38^3 + 48^3  cnt= 2
cube sum= 171288 =   17^3 + 55^3 = 24^3 + 54^3  cnt= 2
cube sum= 195841 =   22^3 + 57^3 = 9^3 + 58^3  cnt= 2
cube sum= 216027 =   3^3 + 60^3 = 22^3 + 59^3  cnt= 2
cube sum= 216125 =   45^3 + 50^3 = 5^3 + 60^3  cnt= 2
cube sum= 262656 =   36^3 + 60^3 = 8^3 + 64^3  cnt= 2
cube sum= 314496 =   30^3 + 66^3 = 4^3 + 68^3  cnt= 2
cube sum= 320264 =   32^3 + 66^3 = 18^3 + 68^3  cnt= 2
cube sum= 327763 =   30^3 + 67^3 = 51^3 + 58^3  cnt= 2
cube sum= 373464 =   54^3 + 60^3 = 6^3 + 72^3  cnt= 2
cube sum= 402597 =   56^3 + 61^3 = 42^3 + 69^3  cnt= 2
cube sum= 439101 =   48^3 + 69^3 = 5^3 + 76^3  cnt= 2
cube sum= 443889 =   38^3 + 73^3 = 17^3 + 76^3  cnt= 2
cube sum= 513000 =   10^3 + 80^3 = 45^3 + 75^3  cnt= 2
cube sum= 513856 =   52^3 + 72^3 = 34^3 + 78^3  cnt= 2
cube sum= 515375 =   54^3 + 71^3 = 15^3 + 80^3  cnt= 2
cube sum= 525824 =   62^3 + 66^3 = 24^3 + 80^3  cnt= 2
cube sum= 558441 =   57^3 + 72^3 = 30^3 + 81^3  cnt= 2
cube sum= 593047 =   63^3 + 70^3 = 7^3 + 84^3  cnt= 2
cube sum= 684019 =   51^3 + 82^3 = 64^3 + 75^3  cnt= 2
cube sum= 704977 =   2^3 + 89^3 = 41^3 + 86^3  cnt= 2
cube sum= 805688 =   30^3 + 92^3 = 11^3 + 93^3  cnt= 2
cube sum= 842751 =   23^3 + 94^3 = 63^3 + 84^3  cnt= 2
cube sum= 885248 =   8^3 + 96^3 = 72^3 + 80^3  cnt= 2
cube sum= 886464 =   12^3 + 96^3 = 54^3 + 90^3  cnt= 2
cube sum= 920673 =   20^3 + 97^3 = 33^3 + 96^3  cnt= 2
cube sum= 955016 =   63^3 + 89^3 = 24^3 + 98^3  cnt= 2
cube sum= 984067 =   35^3 + 98^3 = 59^3 + 92^3  cnt= 2
cube sum= 994688 =   60^3 + 92^3 = 29^3 + 99^3  cnt= 2
cube sum= 1009736 =   59^3 + 93^3 = 50^3 + 96^3  cnt= 2
cube sum= 1016496 =   66^3 + 90^3 = 47^3 + 97^3  cnt= 2
cube sum= 1061424 =   6^3 + 102^3 = 45^3 + 99^3  cnt= 2
cube sum= 1073375 =   23^3 + 102^3 = 60^3 + 95^3  cnt= 2
cube sum= 1075032 =   76^3 + 86^3 = 24^3 + 102^3  cnt= 2
cube sum= 1080891 =   48^3 + 99^3 = 27^3 + 102^3  cnt= 2
cube sum= 1092728 =   1^3 + 103^3 = 64^3 + 94^3  cnt= 2
cube sum= 1195112 =   58^3 + 100^3 = 16^3 + 106^3  cnt= 2
cube sum= 1260441 =   81^3 + 90^3 = 9^3 + 108^3  cnt= 2
cube sum= 1323712 =   40^3 + 108^3 = 76^3 + 96^3  cnt= 2
cube sum= 1331064 =   67^3 + 101^3 = 4^3 + 110^3  cnt= 2
cube sum= 1370304 =   34^3 + 110^3 = 48^3 + 108^3  cnt= 2
cube sum= 1407672 =   14^3 + 112^3 = 63^3 + 105^3  cnt= 2
cube sum= 1533357 =   90^3 + 93^3 = 62^3 + 109^3  cnt= 2
cube sum= 1566728 =   18^3 + 116^3 = 44^3 + 114^3  cnt= 2
cube sum= 1609272 =   55^3 + 113^3 = 92^3 + 94^3  cnt= 2
cube sum= 1728216 =   6^3 + 120^3 = 44^3 + 118^3  cnt= 2
```

```
cube sum= 1729000 =   10^3 + 120^3 = 90^3 + 100^3  cnt= 2
cube sum= 1734264 =   51^3 + 117^3 = 78^3 + 108^3  cnt= 2
cube sum= 1774656 =   93^3 + 99^3 = 36^3 + 120^3  cnt= 2
cube sum= 1845649 =   42^3 + 121^3 = 49^3 + 120^3  cnt= 2
cube sum= 2048391 =   95^3 + 106^3 = 2^3 + 127^3  cnt= 2
cube sum= 2101248 =   16^3 + 128^3 = 72^3 + 120^3  cnt= 2
cube sum= 2301299 =   11^3 + 132^3 = 99^3 + 110^3  cnt= 2
cube sum= 2418271 =   95^3 + 116^3 = 23^3 + 134^3  cnt= 2
cube sum= 2515968 =   60^3 + 132^3 = 8^3 + 136^3  cnt= 2
cube sum= 2562112 =   64^3 + 132^3 = 36^3 + 136^3  cnt= 2
cube sum= 2585375 =   50^3 + 135^3 = 95^3 + 120^3  cnt= 2
cube sum= 2622104 =   60^3 + 134^3 = 102^3 + 116^3  cnt= 2
cube sum= 2691451 =   94^3 + 123^3 = 18^3 + 139^3  cnt= 2
cube sum= 2864288 =   10^3 + 142^3 = 103^3 + 121^3  cnt= 2
cube sum= 2987712 =   12^3 + 144^3 = 108^3 + 120^3  cnt= 2
cube sum= 2991816 =   81^3 + 135^3 = 18^3 + 144^3  cnt= 2
cube sum= 3220776 =   84^3 + 138^3 = 112^3 + 122^3  cnt= 2
cube sum= 3242197 =   76^3 + 141^3 = 85^3 + 138^3  cnt= 2
cube sum= 3375001 =   73^3 + 144^3 = 1^3 + 150^3  cnt= 2
cube sum= 3375008 =   2^3 + 150^3 = 83^3 + 141^3  cnt= 2

Total Taxicab numbers found: 80

Game over, thanks for playing.
ed-vm%
```

For testing, it would be easiest to re-direct the output to a file.