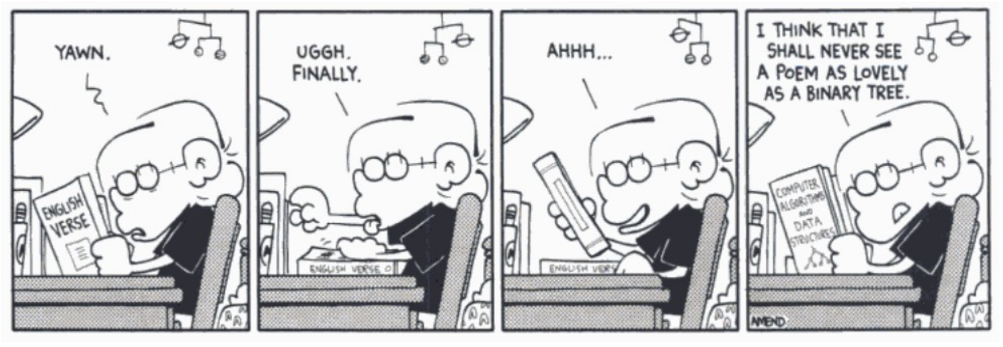
CS 302
Homework, Asst. #09

Purpose: Learn concepts regarding priority queues and heaps.
Due: Thursday (10/30) → Must be submitted on-line before class.
Points: Part A → 75 pts, Part B → 50 pts

Assignment:

Part A:


YAWN.
UGGH. FINALLY.
AHHH...
I THINK THAT I SHALL NEVER SEE A POEM AS LOVELY AS A BINARY TREE.
ENGLISH VERSE
COMPUTER ALGORITHMS AND DATA STRUCTURES
AMEND

Design and implement a C++ class, ***binaryHeap***, to implement a priority queue[1] data structure. The ***binaryHeap*** class will implement a maxHeap priority queue for job scheduling. The job and the priority are provided.

A main will be provided that performs a series of tests. Refer to the UML descriptions for implementation details. The binary tree will be implemented using an array of ***structs***.

Part B:

Create and submit a brief write-up including the following:

- Name, Assignment, Section.
- Summary of the ***priority queue*** data structure.
- Compare the priority queue data structure to using a balanced binary search tree. Include the associated trade-offs.
- The assignment requires using *buildHeap()* instead of calling the *insert()* function multiple times. Explain why and the consequences of doing it incorrectly.
- Big-O for the various priority queue operations (insert, buildHeap, deleteMax, reheapUp, reheapDown, and resize).

Submission:

- Submit a compressed zip file of the program source files, header files, and makefile via the on-line submission by 23:50.
- Submit a copy of the write-up (open document, word, or PDF format).

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*).

Make File:

You will need to develop a make file. You should be able to type:

```
make
```

Which should create the executable.

1 For more information, refer to: http://en.wikipedia.org/wiki/Priority_queue

Test Script:
You will need to develop a simple bash test script. The test script should verify that an executable file is passed as an argument, execute the program with the “-p” option using provided jobs files, capture the output to a file, and compare the program output to a known correct output file (provided). If the files are the same, display Test Successful. If not, display the differences between the files. Refer to the example output to see en example of the output formatting.

Class Descriptions

- Binary Heap Class
  The phone hash class will implement functions specified below.

`binaryHeap`
`-jobElement: struct`
`      -priority: int`
`      -name: string`
`-count: int`
`-heapSize: int`
`-jobHeap: * jobElement`
`+binaryHeap(int=100)`
`+~binaryHeap()`
`+entries() const: int`
`+readJobs(const string): bool`
`+insert(const string, const int): void`
`+deleteMax(string &, int &): bool`
`+isEmpty() const: bool`
`+printJobHeap() const: void`
`-reheapUp(int): void`
`-reheapDown(int): void`
`-buildHeap(): void`
`-resize(): void`

Function Descriptions

- The *binaryHeap()* constructor should initialize the binary heap to an empty state. The parameter must be checked to ensure it is between 100 and 10000 (inclusive). If invalid, the default value should be used.
- The *~binaryHeap()* 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 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 *readJobs()* function should open the passed file name, read the entries into the heap array, and close the file. The function should then call the private *buildHeap()* function to set the heap properties. The function ***must not*** call the *insert()* function multiple times.

- The private *buildHeap()* function to set the heap properties.
- The *deleteMax()* function should remove the maximum 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 funciton should return false and otherwise return the highest priority job information (via reference) and return true.
- The *isEmpty()* function should return true if there are no elements in the heap and false otherwise.
- The *printJobHeap()* function should print the current job heap in level order with a blank line between each level. Refer to the sample output for an example of the formatting.
- 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 tree root and works down to the applicable leaf. Must be written recursively.
- The *resize()* function 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.***

<u>Example Execution:</u>
Below is an example output for the test script and a program execution for the main.

```
ed-vm% ./main -p
****************************************************************
CS 302 - Assignment #9
Binary Heap Job Scheduler

================================================================
Test Set #0
----------------------------------------
Job Heap (level order):
ebay  10

belkin  9
oracle  6

cisco  7
jupiter  8
amazon  2
dell  5

google  1
apple  4
newegg  3

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

```
=================================================================
Test Set #1
----------------------------------------
Job Heap:
LigulaAeneanLLC  100

EgetIpsumIncorporated  97
SitCorporation  68

InterdumCorporation  96
LacusCorp  74
VivamusInstitute  64
SitIndustries  65

CubiliaLLP  67
EuSemInstitute  95
DuiIncorporated  45
InMiPC  50
SemperEgestasInc  63
EtProinCorp  3
EratLLC  26
IntegerPC  23

UltricesCompany  32
ArcuIaculisLtd  27
TortorLtd  10
AliquamCorp  10
FelisOrciConsulting  38
DapibusAssociates  22
UltricesIaculisInc  16
EuSemPellentesqueInc  8

----------------------------------------
Heap Size: 24
Job Priority Order:
LigulaAeneanLLC  100
EgetIpsumIncorporated  97
InterdumCorporation  96
EuSemInstitute  95
LacusCorp  74
SitCorporation  68
CubiliaLLP  67
SitIndustries  65
VivamusInstitute  64
SemperEgestasInc  63
InMiPC  50
DuiIncorporated  45
FelisOrciConsulting  38
UltricesCompany  32
ElitLLP  27
ArcuIaculisLtd  27
EratLLC  26
IntegerPC  23
DapibusAssociates  22
UltricesIaculisInc  16
AliquamCorp  10
TortorLtd  10
EuSemPellentesqueInc  8
EtProinCorp  3

=================================================================
Test Set #2
----------------------------------------
Heap Size: 76
Top 10 Jobs:
MusCompany  100
```

```
TellusLimited  99
FringillaCorp  98
LoremLLP  98
PellentesqueConsulting  98
QuamInc  94
MaurisFoundation  94
VelLLP  93
OrciAdipiscingInc  91
ParturientConsulting  91

=================================================================
Test Set #3
----------------------------------------
Original Heap Size: 205
Test #3, 50 jobs processed...
New Heap Size: 205

Top 10 Jobs (after processing first 50):
AdipiscingElitIndustries  73
MusPC  72
ProinNonMassaFoundation  71
EgestasDuisCompany  71
AdLitoraCompany  70
SitAmetCompany  70
ViverraPC  70
SuspendisseInc  69
HabitantCorporation  69
SitCorporation  68

=================================================================
Test Set #4
----------------------------------------
Original Heap Size: 5000
Test #4, 2000 jobs processed...
New Heap Size: 5000

Top 10 Jobs (after processing first 4000):
Vinte  20
Dynabox  20
Edgeblab  20
Edgeblab  20
Trudeo  20
Brightdog  20
Skynoodle  19
Jetwire  19
Myworks  19
Thoughtbeat  19

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