CPSC 2430 Data Structures

Winter Quarter 2024

Lab 4 - Basic Heap Operations **Due: 10:00 pm, Friday, Feb 9**

This lab is based on the following definition to implement a MIN-HEAP that will enable adoption of the oldest pet (by order of arrival) in our Animal Shelter. We will use the STL **vector** class to store the MIN-HEAP.
Note: You can define a swap function if you think it facilitates your implementation.

```
/*
** Min-Heap construction and basic operations
** Lab 4, CPSC 2430-01
*/
#include <vector>
#include <string>
#include <iostream>

using namespace std;

struct Pet{
  string name;
  int arrival;
};

class PetHeap {
  private:
      vector<Pet*> petHeap;
      void percolateUp(int index);
      void percolateDown(int index);
      void deleteMin();
      Pet* peekMin();
  public:
      PetHeap(){
        petHeap = vector<Pet*>();
      }
      void insert(Pet* pet);
      Pet* adoptOldestPet();
      int numPets();
      void displayPets();
};

// implement the above-mentioned functions below
void PetHeap::percolateUp(int index){
    // maintain heap structure from bottom to top
```

```
    // add your code here
}

void PetHeap::percolateDown(int index){
    // maintain heap structure from top to bottom
    // add your code here
}

void PetHeap::deleteMin(){
    // remove the oldest Pet (by arrival) from the heap
    // and reconstruct the heap after deletion
    // add your code here – calls percolateDown
}

Pet* PetHeap::peekMin(){
    // return the oldest Pet (by arrival)
    // in the heap (don't delete it!)
    // add your code here
}

void PetHeap::insert(Pet* pet){
    // insert a pet into your min heap based on arrival time
    // add your code here – calls percolateUp
}

Pet* PetHeap::adoptOldestPet(){
    // uses peekMin and deleteMin to return adopted pet
    // add your code here
}

int PetHeap::numPets(){
    // return the number of pets in the shelter (heap)
    // add your code here
}

void PetHeap::displayPets(){
    // display the pets (name and arrival time) by level-order
    // add your code here
}
```

You must also have a runTests() function that:

1. inserts 10 pets (arrival time must be random),
2. displays the heap,
3. adopts a few pets,
4. displays number of pets left,
5. adopts all remaining pets,
6. attempts an adoption when no pets are available.

Your lab must also have a user interface to help test each function above individually and quit the program. For example:

```
Welcome to Lab 4 on Basic Heap Operations. What would you like to test?
1. Insert Pet
2. Adopt Pet
3. Number of Pets
4. Display Pets
5. Run Tests
6. Exit
```

After inserting 5 pets with different arrival order (4, 8, 1, 7, 3), a call to Display Pets might look like:

```
The pets in the heap in level order are:
Pusheen 1, Garfield 7, Simba 4, Tigger 8, Felix 3,
```

And after a call to Adopt Pet, the output might look like:

```
Congratulations, you have adopted Pusheen, arrival 1.
```

Similarly for Number of Pets:

```
The shelter heap has 4 pets.
```

Finally, make sure to free any dynamically allocated memory before quitting your program.

Lab 4 Submission

You need to submit the following file:

- lab4.cpp

Your lab4.cpp must include all the functions mentioned above. Before submission, you must ensure your program has been compiled and tested (extensively). Your assignment receives zero if your code cannot be compiled and executed.

You can submit your program multiple times before the deadline. The last submission will be used for grading. To submit your lab, you must execute the following script (assuming your files are on cs1.seattleu.edu):

```
/home/fac/mjilani/submit/24wq2430/lab4_submit
```

Lab 4 Grading Breakdown

Breakdown	Points	Note
User interface	10	You provide a user interface to test your functions
percolateUp	15	Percolate up to maintain the heap property recursively (bottom-up)
percolateDown	20	Percolate down to maintain the heap property recursively (top-down)
insert	10	Insert a pet into your heap
adoptPet	10	Use peekMin and deleteMin to return the adopted Pet
numPets	5	Return the size of your heap
displayPets	5	Display the pets in your heap by level-order
runTests	15	Run tests as described
coding	10	Your code is clean and well commented; all allocated memory is released before quitting the program. Proper functional decomposition, no use of global variables, etc.