Class CMPS 261
Section 001
Problem Programming Assignment #2
Name McKelvy, James Markus
CLID Jmm0468
Due Date 12:30pm October 20, 2005

# II. Design Documentation

## II.1 System Architecture Description

The object(s) within the program are:
minHeap: minHeap.h

The main objective of the application is to utilize the implementation of the minHeap class. The application's main driver will be used to test the overall stability and integrity of the implementation of the minHeap class by testing its creation and its member functions. The main driver will create only one kind of heap: an empty minimum heap.

## II.2 Information about the Objects

### II.2.1 Class Information
Name: minHeap
Description: Simulates a minimum heap with the ability to have items inserted into it and its minimum element removed.
Base Class: N/A

### II.2.2 Class Attributes
Name: heap
Description: This is a pointer to an array of elements that represent the minimum heap tree.
Type: Type *
Range of acceptable values: Any value that corresponds to the template class “Type”.

Name: count
Description: Keeps track of the current number of elements in the heap.
Type: int
Range of acceptable values: Any number greater than or equal to zero.

Name: maxSize
Description: Keeps track of the maximum size of this heap
Type: int
Range of acceptable values: Any number greater than zero.

### II.2.3 Class Operations
Prototype: minHeap(int size);
Description: Default Constructor
Precondition: There is enough memory to be allocated

Postcondition: Creates an empty minimum heap with a max size of maxSize if and only if maxSize > 0, else the max size will be set to a default value of 15.
Cost Analysis: O(1)
Visibility: public

Prototype: ~minHeap()
Description: Destructor
Precondition: The minimum heap has been allocated
Postcondition: deallocates the memory used by this object
Cost Analysis: O(1)
Visibility: public

Prototype: bool insert(Type item);
Description: Inserts the item into the heap.
Precondition: The heap must be initialized and the item must be of type Type.
Postcondition: The item is inserted to the heap and is moved to the correct level in the tree.
Cost Analysis: log(n)
Visibility: public

Prototype: Type removeMin();
Description: Removes the minimum item from the heap
Precondition: The heap must be initialized and have an item to remove.
Postcondition: The minimum item is removed from the heap (if it exists) and returned, otherwise null is returned.
Cost Analysis: log(n)
Visibility: public

Prototype: bool isLeaf(int index);
Description: Checks to see if the current index is a leaf in the heap.
Precondition: The heap must be initialized.
Postcondition: Returns true if the item is a leaf node, false otherwise.
Cost Analysis: O(1)
Visibility: private

Prototype: int leftChild(int index);
Description: Finds the position of the left child from the given index.
Precondition: The index must be valid and the heap must be initialized
Postcondition: Returns the index of the left child (if it exists), or a -1 otherwise.
Cost Analysis: O(1)
Visibility: private

Prototype: int rightChild(int index);
Description: Finds the position of the right child from the given index.
Precondition: The index must be valid and the heap must be initialized
Postcondition: Returns the index of the right child (if it exists), or a -1 otherwise.
Cost Analysis: O(1)

Visibility: private

Prototype: int parent(int index);
Description: Finds the position of the parent from the given index.
Precondition: The index must be valid and the heap must be initialized
Postcondition: Returns the index of the parent (if it exists), or a -1 otherwise.
Cost Analysis: O(1)
Visibility: private

II.3 Information about the Main Application

```
#include<iostream>
#include<cstdlib>
#include "minHeap.h"

using namespace std;

// Prototype: void pause();
// Description: Pauses for the user.
// Precondition: None.
// Postcondition: Continues execution of the program.
// Cost Analysis: O(1)
// Visibility: public
void pause();

int main(){
   int temp = -1;
   int numset1[] = {1,2,3,4,5,6,7,8,9,10};
   int numset2[] = {10,9,8,7,6,5,4,3,2,1};
   int numset3[] = {3,4,2,1,5,7,6,9,8,10};
   int numset4[] = {5,4,3,2,1,6,7,8,9,10};
   int numset5[] = {34, 21, 35, 98, 32, 45, 12, 87, 76, 65, 52, 100, 223, 9, 214};
   int numset6[] = {2, 7, 4, 13, 1983, -1, 76, 3, 13, 27, 49, 44, 550, 261, 134, 543,
                    246, 86, 5556, 2005, 58, -4, 64, 667};

   // set 1
   cout << "Creating a minHeap object with a size of 10." << endl;
   minHeap<int> * h1 = new minHeap<int>(10);

   cout << "Inserting the following elements into the heap:" << endl << endl;
   for(int i = 0; i < 10; i++){
     cout << " " << numset1[i];
   }
   cout << endl << endl;
   for(int i = 0; i < 10; i++){
     h1->insert(numset1[i]);
   }
```

```
cout << "Removing all elements from the heap: " << endl << endl;
while(temp != NULL){
  temp = h1->removeMin();
  if(temp != NULL)
    cout << " " << temp;
}
cout << endl;
temp = -1;
delete h1;

pause();

// set 2
cout << "Creating a minHeap object with a size of 10." << endl;
minHeap<int> * h2 = new minHeap<int>(10);

cout << "Inserting the following elements into the heap:" << endl << endl;
for(int i = 0; i < 10; i++){
  cout << " " << numset2[i];
}
cout << endl << endl;
for(int i = 0; i < 10; i++){
  h2->insert(numset2[i]);
}

cout << "Removing all elements from the heap: " << endl << endl;
while(temp != NULL){
  temp = h2->removeMin();
  if(temp != NULL)
    cout << " " << temp;
}
cout << endl;
temp = -1;
delete h2;

pause();

// set 3
cout << "Creating a minHeap object with a size of 10." << endl;
minHeap<int> * h3 = new minHeap<int>(10);

cout << "Inserting the following elements into the heap:" << endl << endl;
for(int i = 0; i < 10; i++){
  cout << " " << numset3[i];
}
cout << endl << endl;
for(int i = 0; i < 10; i++){
  h3->insert(numset3[i]);
```

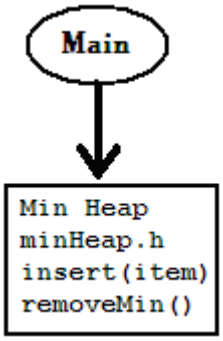
```
}

cout << "Removing all elements from the heap: " << endl << endl;
while(temp != NULL){
  temp = h3->removeMin();
  if(temp != NULL)
    cout << " " << temp;
}
cout << endl;
temp = -1;
delete h3;

pause();

// set 4
cout << "Creating a minHeap object with a size of 10." << endl;
minHeap<int> * h4 = new minHeap<int>(10);

cout << "Inserting the following elements into the heap:" << endl << endl;
for(int i = 0; i < 10; i++){
  cout << " " << numset4[i];
}
cout << endl << endl;
for(int i = 0; i < 10; i++){
  h4->insert(numset4[i]);
}

cout << "Removing all elements from the heap: " << endl << endl;
while(temp != NULL){
  temp = h4->removeMin();
  if(temp != NULL)
    cout << " " << temp;
}
cout << endl;
temp = -1;
delete h4;

pause();

// set 5
cout << "Creating a minHeap object with a size of 14." << endl;
minHeap<int> * h5 = new minHeap<int>(14);

cout << "Inserting the following elements into the heap:" << endl << endl;
for(int i = 0; i < 15; i++){
  cout << " " << numset5[i];
}
cout << endl << endl;
```

```
    for(int i = 0; i < 15; i++){
      h5->insert(numset5[i]);
    }

    cout << "Removing all elements from the heap: " << endl << endl;
    while(temp != NULL){
      temp = h5->removeMin();
      if(temp != NULL)
        cout << " " << temp;
    }
    cout << endl;
    temp = -1;
    delete h5;

    pause();

    // set 6
    cout << "Creating a minHeap object with a size of 22." << endl;
    minHeap<int> * h6 = new minHeap<int>(22);

    cout << "Inserting the following elements into the heap:" << endl << endl;
    for(int i = 0; i < 22; i++){
      cout << " " << numset6[i];
    }
    cout << endl << endl;
    for(int i = 0; i < 24; i++){
      h6->insert(numset6[i]);
    }

    cout << "Removing all elements from the heap: " << endl << endl;
    while(temp != NULL){
      temp = h6->removeMin();
      if(temp != NULL)
        cout << " " << temp;
    }
    cout << endl;
    temp = -1;
    delete h6;

    pause();

    return 0;
}

void pause(){
  cout << "Press ENTER.";
  getchar();
  cout << endl << "----------------------------" << endl;
```

}

II.4 Design Diagrams

II.4.1 Object Interaction Diagram




II.4.2 Aggregation Diagram