There are seven (7) questions on this exam over 6 pages. It is okay to write helper function(s) for any of the questions on this exam.
The definitions of data-types that you need for some of these questions are on page 7. Please do not hesitate to ask questions.

Problem 1 (10 points)
Insert the following values into a binomial queue. Show your work at every step.

40, 50, 10, 5, 100, 200, 150, 60, 45, 210, 35, 74, 80, 14, 15

Now, perform a delete-min on the resulting heap.
Problem 2 (10 points)
Insert the following values into a heap-ordered tree that is being implemented using an array. Show your work.

40, 50, 10, 5, 100, 200, 150, 60, 45, 210, 35, 74, 80, 14, 15

Now, perform a delete-min on the resulting heap.

Problem 3 (10 points)
How do you find the \( k^{th} \) smallest value in a heap-ordered tree? Explain.
Problem 4 (10 points)

Insert the following values into a hash-table with capacity 17 (with indices 0 through 16, inclusive) using the hash function, \( f(x) = x \% 17. \)

10, 78, 105, 215, 64, 420, 129, 155, 16, 33

Resolve the collisions using:

1. Linear Probing.

2. Chaining.

The following table helps with the computation of modular functions that you need to answer this question.

<table>
<thead>
<tr>
<th>( x )</th>
<th>( x % 17 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>78</td>
<td>2</td>
</tr>
<tr>
<td>105</td>
<td>3</td>
</tr>
<tr>
<td>215</td>
<td>11</td>
</tr>
<tr>
<td>64</td>
<td>13</td>
</tr>
<tr>
<td>420</td>
<td>12</td>
</tr>
<tr>
<td>129</td>
<td>10</td>
</tr>
<tr>
<td>155</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>33</td>
<td>16</td>
</tr>
</tbody>
</table>
Problem 5 (10 points)

Write an implementation for a circular queue that stores the elements in an array. Use this class for this purpose.

class CircularQueue {
    public:
        CircularQueue();
        void insert(int v);
        void remove();

    private:
        int queue[100], head, tail;
};

*insert* adds the given element to the queue at one end and *remove* deletes one element from the other end; the same behavior that you observed when implementing the queue that held the segments of the *worm* in your last lab. The queue will never contain more than 99 elements at one time. In addition, *remove* will not be called when the queue is empty.

Write an implementation for the two functions, *insert* and *remove*. These operations should run in constant time.
Problem 6 (25 points)

This question is based on the make project. We have parsed and built a graph using the 
GraphNode.hpp that you were given for the make project. We want to find the node of the 
graph that has the maximum number of children. That is, the node that represents the target 
that is dependent on more items than any other target. Assume that the graph is acyclic. The 
definition of GraphNode.hpp appears on the last page.

GraphNode *nodeWithMaxItems(GraphNode *gNode)  
// gNode is a pointer to GraphNode whose definition is given to you.  
// It points to the root-node (the target to be made). This function  
// returns a pointer to the node that has the maximum number of children.  
// The graph is acyclic.
Problem 7 (25 points)

Using an array that contains $n$ ordered values, build a binary-search tree that is reasonably balanced. That is, a tree that has roughly the same number of nodes in its left- and right-subtrees. The definition of `TreeNode` to be used for this problem is attached to this exam.

```c
TreeNode *buildTree(int a[], int n)
/*
   "a" contains "n" sorted values. Using these values, build a binary-search
tree that is reasonably balanced. That is, a tree that has roughly the
same number of nodes in its left and right subtrees.
*/
```