Computer Science 302 Spring 2017 (Practice for) Final Examination, May 10, 2017

Name:_____

The entire practice examination is 1005 points.

1. True or False. [5 points each]

   (a) _____ The time to heapsort an array of n items is $O(n \log n)$.

   (b) _____ Open hashing uses open addressing.

   (c) _____ In the decision tree model of computation, the time complexity of any algorithm to sort n items is $\Omega(n \log n)$.

   (d) _____ The height of a binary tree with n nodes is $O(\log n)$.

   (e) _____ A binary search tree is commonly used to represent unfulfilled obligations.

   (f) _____ Quicksort takes $O(n \log n)$ time to sort an array of n items.

   (g) _____ Given the choice between two algorithms, one of which takes $O(n)$ time and the other of which takes $O(n^2)$ time, it is *always* best to choose the one which takes $O(n)$ time.

   (h) _____ The built-in function `random` is an excellent choice for a hash function.

   (i) _____ A queue is typically used to store unfulfilled obligations.

   (j) _____ A stack is typically used to store unfulfilled obligations.

   (k) _____ A heap is typically used to store unfulfilled obligations.

   (l) _____ A hash table is typically used to store unfulfilled obligations.

2. [10 points] What search structure should you use if the average number of items that will be in the structure at any given time is two?

3. [5 points] What are the smallest and largest possible number of nodes that a 2-3 tree with height 3 could have? _____ and _____.

4. [5 points] You cannot use _____ algorithm if any edge has negative weight.

5. [10 points] What implementation of the ADT search structure would you use if the expected number of items in the structure is 1?

6. [5 points] Push and pop are operators of the ADT _____.

7. [10 points]_____ and _____ are operators of the ADT *array*.

8. [30 points]Each of the following answers is from this list: *greedy*, *divide and conquer*, *dynamic programming*.

   (a) How would you describe Kruskal's algorithm?

   (b) How would you describe mergesort?

   (c) How would you describe quicksort?

   (d) How would you describe binary search?

9. [10 points each] For each of the following code fragments, express the asymptotic time complexity by choosing the best of the following answers: $O(n)$, $O(n^2)$, $O(n \log n)$, $O(\log n)$, $O(\log \log n)$, $\Theta(n)$, $\Theta(n^2)$, $\Theta(n \log n)$, $\Theta(\log n)$, $\Theta(\log \log n)$

   (a)
```
for (int i = 0; i < n; i++)
  cout << "Hi there.";
```

   (b)
```
for (int i = 0; i < n; i = 2*i+1)
  cout << "Hi there.";
```

   (c)
```
for (int i = 0; i < n; i++)
  for (int j = i; j > 0; j = j/2);
  cout << "Hi there.";
```

   (d)
```
for (int i = 0; i < n; i++)
  for (int j = n; j > i/2; j = j/2);
    cout << "Hi there.";
```

   (e)
```
for (int i = 0; i < n; i = i*i+1)
  cout << "Hi there.";
```

10. [30 points]

    (a) Describe the meaning of the word collision as used in discussions of hashing.

    (b) How are collisions handled in closed hashing?

    (c) How are collisions handled in open hashing?

11. [10 points] What implementation of the ADT search structure would you use if n items are to be inserted at once at the beginning of the program, there will be no further inserts, and find will be executed n^2 times during the running of the program? (There is more than one correct answer to this problem, as well as several inferior answers.)

12. [20 points] Walk through the steps of the stack algorithm used to evaluate the following postfix expression, showing the stack at each step: (Hint: there will be approximately 9 illustrations of the stack.)

```
5 6 + 3 * 2 3 * -
```

13. [20 points] Find an optimal prefix code for the alphabet $\{A, B, C, D, E, F, G, H\}$, if the frequencies of the symbols are as given in the following table:

A	35
B	7
C	32
D	5
E	16
F	4
G	11
H	5

14. [30 points]

    The *Partition* step of Quicksort has a loop invariant. Give that loop invariant, and illustrate its meaning by drawing a figure, or figures.

15. [30 points] Describe each of the following types of search. (Be sure to say what the structure is that is being searched in each case.)

    (a) Linear search.

    (b) Binary search.

16. [20 points] Write pseudocode for the array implementation of the ADT "stack of integer." Your code should include procedures that implement **pop**, **push**, and **empty**.

17. [20 points] Explain how you would insert and delete from a stack, given that you are using singly linked nodes. Draw pictures.

18. [25 points] Explain how insertion works in a 2,3-tree. Hint: the phrase "node splitting" or the equivalent must be in your explanation.

19. [30 points] Given the following:

```
class BST{ // Binary Search Tree
  public:
   BST(int); // initializes item field to parameter, links to 0
   void static postorderWrite(BST*); // all items to standard ostream inorder
   void static insert(int, BST*&); // inserts parameter, if not there
   bool find(int, BST*); // parameter is in the binary search tree
  private:
   int item; // the value stored in the node
   BST * left; // pointer to the left subtree
   BST * right; // pointer to the right subtree
 };
```

Complete the following code by writing **exactly three lines:**

```
     void BST::postorderWrite(BST * t){
        if (t != 0){
// Your three lines go here.


          }
       }
```

20. [30 points] Suppose you are writing a dynamic programming algorithm to find the minimum weight path between a given source vertex S and a given target vertex T in a weighted directed acyclic graph G.

    (a) Describe the subproblems.

    (b) In what order would you work the subproblems?

21. True or False. [5 points each]

    (a) _____ Quicksort takes $O(n \log n)$ expected time to sort an array of n items, provided randomization is used to pick the pivot items.

    (b)

22. [20 points] Draw "before" and "after" figures illustrating *left rotation*.

23. True or False. [5 points each]

    (a) _____ The time to heapsort an array of n items is $\Theta(n \log n)$.

    (b) _____ Open hashing uses open addressing.

    (c) _____ The height of a binary tree with n nodes is $\Omega(\log n)$.

24. [10 points] What implementation of the ADT search structure would you use if n items are to be inserted at once at the beginning of the program, there will be no further inserts, and find will be executed n^2 times during the running of the program? (There is more than one correct answer to this problem, as well as several inferior answers.)

25. [20 points] Explain how you would insert and delete from a stack, given that you are using the standard array implementation of stack.

26. [30 points] Describe each of the following types of search. (Be sure to say what the structure is that is being searched in each case.)

(a) Breadth first search.

(b) Depth first search.

27. [40 points] Give four ways to implement the ADT "search structure." (For example, "binary search tree" is one way, so don't use that one.) Just give the names of the implementations, not any details.

28. [10 points] You can sort a set of items using a binary search tree, as follows: Start with an empty binary search tree, insert the items one at a time, then visit the nodes of the tree in inorder, writing out the items. This algorithm is essentially the same as which one of these well-known sorting algorithms? (Choose one answer.)

(a) Quicksort

(b) Heapsort

(c) Mergesort

29. [10 points each] For each of the following code fragments, express the asymptotic time complexity by choosing the best of the following answers: zero, $O(n)$, $O(n^2)$, $O(n \log n)$, $O(\log n)$, $O(\log \log n)$, $\Theta(n)$, $\Theta(n^2)$, $\Theta(n \log n)$, $\Theta(\log n)$, $\Theta(\log \log n)$, infinite loop: never ends.

(a)
```
for (int i = 0; i < n; i := i+2)
  cout << "Hi there.";
```

(b)
```
for (int i = 0; i < n; i = 2*i)
  cout << "Hi there.";
```

(c)
```
for (int i = 0; i < 0; i++)
  for (int j = i; j > 0; j = j/2);
  cout << "Hi there.";
```

(d)
```
for (int i = 0; i < n; i++)
  for (int j = i; j > i/2; j = j/2);
    cout << "Hi there.";
```

(e)
```
for (int i = 1; i < n; i = i*i)
  cout << "Hi there.";
```

30. [20 points] Write C++ code for the **find** portion of union-find. Be sure to use path compression. Do not include any other part of the program. If you write more than 10 lines, you've written far too much.

31. [30 points] Given the following:

```
class BST{ // Binary Search Tree
  public:
   BST(int); // initializes item field to parameter, links to 0
   void static preorderWrite(BST*); // all items to standard ostream inorder
   void static insert(int, BST*&); // inserts parameter, if not there
```

```
    bool find(int, BST*); // parameter is in the binary search tree
  private:
    int item; // the value stored in the node
    BST * left; // pointer to the left subtree
    BST * right; // pointer to the right subtree
  };
```

Complete the following code by writing **exactly three lines:**

```
    void BST::preorderWrite(BST * t){
       if (t != 0){
// Your three lines go here.


        }
     }
```

32. [10 points each] For each of the following code fragments, express the asymptotic time complexity by choosing the best of the following answers: $O(n)$, $O(n^2)$, $O(n \log n)$, $O(\log n)$, $O(\log \log n)$.

(a)
```
    for (int i = 0; i < n; i++)
      cout << "Hi there.";
```

(b)
```
   for (int i = 0; i < n; i = 2*i+1)
      cout << "Hi there.";
```

(c)
```
    for (int i = 0; i < n; i++)
      for (int j = i; j > 0; j = j/2);
      cout << "Hi there.";
```

(d)
```
    for (int i = 0; i < n; i++)
      for (int j = i; j > i/2; j = j/2);
        cout << "Hi there.";
```

(e)
```
   for (int i = 0; i < n; i = i*i+1)
      cout << "Hi there.";
```

33. [30 points]

(a) Describe the meaning of the word *collision* as used in discussions of hashing.

(b) How are collisions handled in closed hashing?

(c) How are collisions handled in open hashing?

34. [10 points] What implementation of the ADT search structure would you use if n items are to be inserted at once at the beginning of the program, there will be no further inserts, and find will be executed n^2

times during the running of the program? (There is more than one correct answer to this problem, as well as several inferior answers.)

35. [20 points] Explain how you would implement a sparse array using a search structure. Do **not** give any details whatsoever about the search structure itself, since that's not the point of this question.

36. [20 points] Explain how you would insert and delete from a queue, given that you are using singly linked nodes in a circular linked list implementation. Draw pictures.

37. [30 points] Given the following:

```
class BST{ // Binary Search Tree
  public:
   BST(int); // initializes item field to parameter, links to 0
   void static inorderWrite(BST*); // all items to standard ostream inorder
   void static insert(int, BST*&); // inserts parameter, if not there
   bool find(int, BST*); // parameter is in the binary search tree
  private:
   int item; // the value stored in the node
   BST * left; // pointer to the left subtree
   BST * right; // pointer to the right subtree
 };
```

Complete the following code by writing **exactly three lines:**

```
    void BST::inorderWrite(BST * t){
       if (t != 0){
// Your three lines go here.


        }
     }
```

38. [30 points] Use polyphase mergesort to sort the following list: A,L,G,O,R,I,T,H,M,S,R,F,U,N. Show all steps.

39. [20 points] Let H be a trep, where the alphabetically smallest item is at the root. Execute the following commands, showing the treap at each step.

    (a) Insert(Bob,5)

    (b) Insert(Tim,3)

    (c) Insert(Ann,1)

    (d) Delete(Ann)

40. [30 points]

    (a) Under what conditions could you use dynamic programming to solve the single source minimum path problem?

    (b) Write pseudocode for dynamic programming for the single source minimum path problem.

41. [30 points]

    (a) Under what conditions could you use the Bellman-Ford algorithm?

    (b) Write pseudocode for the Bellman-Ford algorithm.

42. [30 points]

    (a) Under what conditions could you use the Floyd-Warshall algorithm?

    (b) Write pseudocode for the Floyd-Warshall algorithm.