CISS350: Data Structures and Advanced Algorithms
Final exam f01

Name: ______________________________ Score: $\boxed{}$

Open `main.tex` and enter answers (look for `answercode`, `answerbox`, `answerlong`). Turn the page for detailed instructions. To rebuild and view pdf, in bash shell execute `make`. To build a gzip-tar file, in bash shell execute `make s` and you'll get `submit.tar.gz`.

Honor Statement

I, $\boxed{\text{[REPLACE WITH YOUR FULLNAME]}}$, attest to the fact that the submitted work is my own and is not the result of plagiarism. Furthermore, I have not aided another student in the act of plagiarism.

Question	Points
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	

Question	Points
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	
32	
33	
34	
35	
36	
37	
38	
39	
40	

TOTAL	

This is a

- Closed-book

- Closed-notes
- No calculator
- No computer (except your brain)
- No discussion (except with yourself)

Write legibly.

How to describe a binary tree in text

For the above questions, here's an example on how to describe a binary tree in text. For this tree

```
            (10)
          /      \
      ( 4)        (25)
     /           /    \
 ( 1)        (12)      (27)
                           \
                            (29)
```

you write

```
[10, [4, 25]]
[4, [1, None]]
[25, [12, 27]]
[1, [None, None]]
[12, [None, None]]
[27, [None, 29]]
[29, [None, None]]
```

Q1. [Derive big–O of runtime of a given algorithm]

The following computes the sum of squares from 1^2 to n^2:

```
s = 0
for i = 0, ..., n - 1:
    term = i * i
    s = s * s + term
```

Here's the program with goto statements and timing for each statement:

```
         i = 1                time t0
         s = 0                time t1
LOOP:    if i > n:            time t2
             goto ENDLOOP     time t3
         term = i * i         time t4
         s = s + term         time t5
         i = i + 1            time t6
         goto LOOP            time t7
ENDLOOP:
```

(a) Compute the time taken as a function of n. The expression should contain constants t_0, ..., t_7.
Answer:

$T(n) =?$

(b) Simplify the runtime function by giving names A, B, ... to the constants of the function from (a).
Answer:

$T(n) =?$

(c) Fudge away the constants and write down the simplest $g(n)$ such that the time in (b) is a big-O of your $g(n)$. Your $g(n)$ should be either n^0 (i.e., 1) or n or n^2 or n^3 or ...
Answer:

$T(n) =?$

Q2. [Recognize big–O of algorithm]

(a) What is the big–O of the runtime of the following algorithm:

```
s = 0
for i = 0, 1, 2, ..., n - 1:
    for j = 0, 1, 2, ..., i:
        for k = j, j + 1, ..., n - 1:
            s = s + x[i][j] + k
```

Answer:

$T(n) =?$

(b) What is the big–O of the runtime of the following algorithm:

```
s = 0
for i = 0, 1, 2, ..., n - 1:
    for j = 0, 1, 2, 3::
        for k = i, i + 1, ..., n - 1:
            s = s + x[i][k] * j
```

Answer:

$T(n) =?$

(c) What is the big–O of the best runtime of the following algorithm:

```
s = 0
for i = 0, 1, 2, ..., n - 1:
    if x[0][0] > 0:
        for j = 0, 1, 2, 3::
            for k = i, i + 1, ..., n - 1:
                s = s + x[i][k] * j
```

Answer:

$T(n) =?$

Q3. [Recall big-O runtime performance of standard algorithms]

For questions on runtime, just state the big-O of the runtime performance.

(a) What is the worst runtime of mergesort?
Answer:

$T(n) = ?$

(b) What is the best runtime of mergesort?
Answer:

$T(n) = ?$

(c) What is the average runtime of mergesort?
Answer:

$T(n) = ?$

(d) What is the worst runtime of quicksort?
Answer:

$T(n) = ?$

(e) What is the best runtime of quicksort?
Answer:

$T(n) = ?$

(f) What is the average runtime of quicksort?
Answer:

$T(n) = ?$

(g) What is the worse runtime of heapsort?
Answer:

$T(n) = ?$

Q4. [Trace of basic sorting algorithms]

Perform selection sort on the following array, showing the result after each pass:

```
{5, 1, 2, 4, 3, 6}
```

Answer:

```
{5, 1, 2, 4, 3, 6}
```

Q5.

(a) What are the correct intermediate steps of the following

$$15, 20, 10, 18$$

when it is being sorted with the quicksort? Write down *all* that applies. (Each stage is the array after pivoting and partitioning. There are of course many methods for choosing a pivot and many ways to partition – you do not need to know the specific method to answer this question.)

(A) 15,10,20,18 → 15,10,18,20 → 10,15,18,20
(B) 15,20,10,18 → 15,10,20,18 → 10,15,20,18
(C) 10,20,15,18 → 10,15,20,18 → 10,15,18,20
(D) 10,20,15,18 → 10,18,15,20 → 10,15,18,20
(E) None of the above.

Answer:

(b) Write down *all* that applies: Quicksort uses

(A) finding maximum
(B) partitioning
(C) finding a pivot
(D) merging

Answer:

(c) Write down one of the two: Quicksort is a stable sort.

(A) True
(B) False

Answer:

Q6. You are given the following singly linked node class:

```
class Node
{
public:
    Node(int key=0, Node * next=NULL)
        : key_(key), next_(next)
    {}

    int key_;
    Node * next_;
};
```

Note that all members are public (to make life simple). The following is a class for singly linked list (of integers):

```
class SLList
{
public:
    SLList()
        : phead_(NULL)
    {}

    // other methods

private:
    Node * phead_;
};
```

Complete the following methods that removes the head node if the linked list is not empty. If the linked list is empty (i.e., `phead_` is `NULL`), nothing is done (i.e., do not throw any exception). Note that the return type is `void`, i.e., do not return any value.
Answer:

```
void SLList::delete_head()
{









}
```

Q7. [Doubly linked list]

You are given an incomplete skeleton of a doubly linked list implementation:

```
class Node
{
public:
    Node(int key=0, Node * prev=NULL, Node * next=NULL)
        : key_(key), prev_(prev), next_(next)
    {}
    int key_;
    Node * prev_;
    Node * next_;
};

class DLList
{
public:
    DLList()
        : pheadsentinel_(new Node(0, NULL, NULL)),
          ptailsentinel_(new Node(0, NULL, NULL))
    {
        pheadsentinel_->next_ = ptailsentinel;
        ptailsentinel_->prev_ = pheadsentinel;
    }
private:
    Node * pheadsentinel_;
    Node * ptailsentinel_;
};
```

Complete the following method to perform insert tail.

Answer:

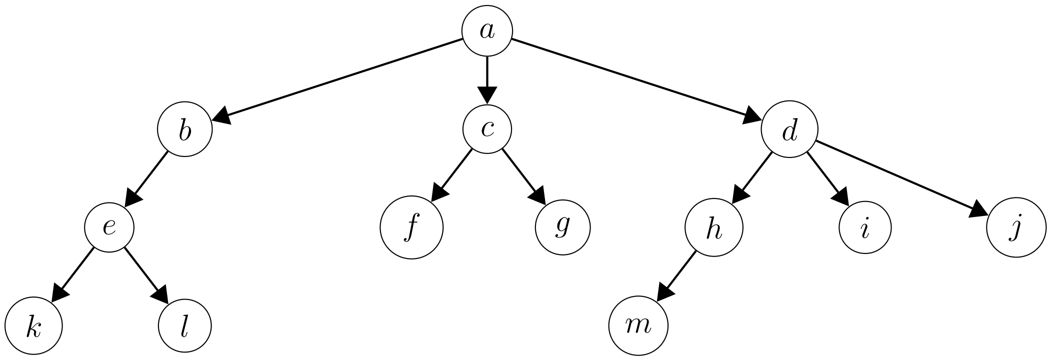
```
void DLList::insert_tail(int key)
{





}
```

Q8. [Tree]



(a) List the values of the nodes using pre-order traversal:

Answer:

(b) List the values of the nodes using post-order traversal:

Answer:

(c) What is the height of the tree?

Answer:

(d) What is the depth of h?

Answer:

(e) What are the siblings of i?

Answer:

(f) Suppose `Node` is a binary tree node class. Assume that the class has `get_left()` and `get_right()` methods to return the left and right pointer of a `Node` object. Complete the following function that computes the height at `p`. You may assume that there's a `max()` function such that `max(a, b)` returns the maximum of `a` and `b`.

Answer:

```
int height(const Node * const p)
{

}
```

Q9. [BST]

Write down the runtimes of the following operations on a BST (binary search tree) that has n nodes.

(a) Best runtime for insert (when given key):
Answer:

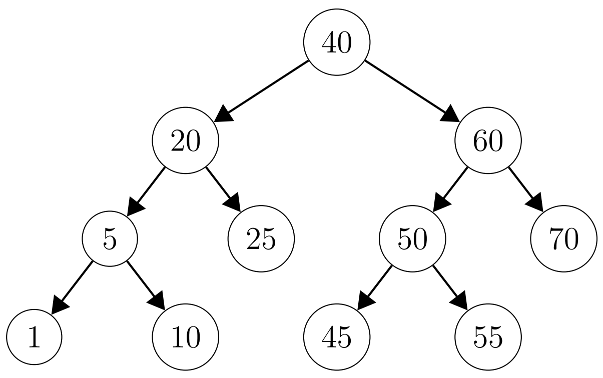
(b) Worst runtime for insert (when given key):
Answer:

(c) Best runtime for delete (when given key):
Answer:

(d) Worst runtime for delete (when given key):
Answer:

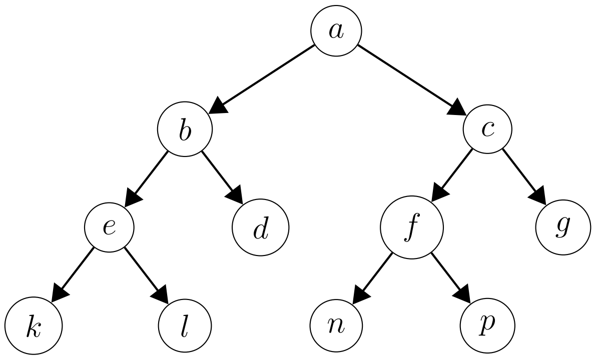
(e) Worst runtime for search (returning pointer when given key):
Answer:

(f) You are given this BST:



Describe the resulting BST in text after deleting the value 20. (Recall our choice of "always using left subtree whenever possible" when doing delete.)
Answer:

Q10. [Balancing tree] Let T be the following tree:



(a) Describe the tree T in text after a right rotation at c.
Answer:

(b) Draw the tree T after a left rotation at a. (Note: T refers the tree at the beginning of the question and not the resulting tree of (a).)
Answer:

(c) Going back to the original tree T in (a), suppose q was inserted on the left of k (and the tree was re-balanced if necessary), followed by inserting r on the right of q (and the tree was re-balanced if necessary), followed by inserting s on the ~~left of r~~ right of q (and the tree was re-balanced if necessary), Describe the resulting tree in text.
Answer:

(d) Going back to the original tree T in (a), suppose q was inserted on the right of k (and the tree was re-balanced if necessary) followed by inserting r on the left of q (and the tree was re-balanced if necessary). Describe the resulting tree in text.
Answer:

Q11. [Heaps]

Starting with an empty max heap (as an array of course), perform the following operations. Show the result of each step after each heapify operation. Describe the heap in text like an array such as `{1, 2, 3, 4, 5}` Note that the heap is a *max* heap.

(a) Insert 5:
ANSWER:

```
{5}
```

(a) Insert 1:
ANSWER:

```
{}
```

(b) Insert 2:
ANSWER:

```
{}
```

(c) Insert 4:
ANSWER:

```
{}
```

(d) Insert 3:
ANSWER:

```
{}
```

(e) Insert 6:
ANSWER:

```
{}
```

(f) Delete max (the root):
ANSWER:

```
{}
```

(g) Delete max (the root):
ANSWER:

```
{}
```

Q12. [Heapsort]

(a) Consider the following array:

`{5, 1, 0, 2, 3, 7, 3, 4}`

Perform operations on the array to build a maxheap. Draw the array after *every* swap.

Answer:

```
{5, 1, 0, 2, 3, 7, 3, 4}
```

(b) You are given a maxheap (as an array). Perform heapsort on the array, drawing the array after *every* swap.

```
{7,5,6,3,2,4,0,1}
```

ANSWER:

```
{7, 5, 6, 3, 2, 4, 0, 1}
```

Q13. [Hash table]

Let the following function

$$h(x) = 2x + 1$$

be used as a hash function.

(a) In a hash table of size 8, insert the keys 1,3,2,5,4,8,6,7. You must insert the keys in the given order, going left to right. (Of course it the hash value is too huge, you mod is by the size of the table.) Draw the table below. The first number for each row is the index of that row, the second is for a flag that indicates the state of the row:

- If a row is not occupied, the flag value is Available.
- If a row is occupied, the flag value is Not-Available.
- If a row was previously occupied (but that row was then deleted), the flag value is Deleted.

The third column is for the key. The fourth column is for the value of a row and can be left blank.

As an example, if the index 0 row is occupied by key 99, then the row at index 0 is

```
0: {Not-Available, 99,}
```

See table below for this example.

In the event of a hash collision, the collision resolution method you should use is linear probe, i.e., the "next row" method.

Answer:

```
0: {Not-Available, 99, }
1: { , , }
2: { , , }
3: { , , }
4: { , , }
5: { , , }
6: { , , }
7: { , , }
```

(The data at index 0 is only an example. Correct it.)

(b) Continue with (a). For each of the following keys, write down the number of rows accessed in order to find that key. Write down the average number of rows accessed.

Answer:

```
key 1:
key 2:
key 3:
```

```
key 4:
key 5:
key 6:
key 7:
key 8:
Average:
```

(c) Continue with (b). What is the worst case in terms of number of rows accessed during a find of a key *not* in the table?
Answer:

Q14. [Applications of linked list]

(a) Convert the following infix arithmetic expression

$$2 * 1 - 3/(6 + 5 - 4) + 7$$

into RPN. (Do *not* simplify, i.e., all the above digits must appear in the RPN expression.)

Answer:

(b) Evaluate the following arithmetic expression written in RPN

```
3 4 + 7 2 1 + / - 6 5 * -
```

Answer:

(c) In part (b), what is the maximum size of the stack during the computation?

Answer:

Q15. (a) Describe an algorithm that you would use to sort an array `x` of 1000000000 integers with values in the following ranges: each `x[i]` is in -100,...,-91 or in 1000000, ..., 1000009.

Answer:

(b) Next, provide a C++ implementation. Of course assume you want the fastest algorithm. Your answer is incorrect if your runtime is not the best possible.

Answer:

```
void sort(int x[])
{
    int t[           ]; // If your sorting is in-place, delete this line


}
```

Q16. Implement mergesort to sort an array of integer values in ascending order. The prototype is `mergesort(int * start, int * end, int * t)`. The array to be sorted is `*start`, ..., `*(end - 1)`. You may assume `t` points to a temporary array that is large enough for your mergesort. (You probably want a merge function.)

Answer:

Q17. Suppose you are given the following node class for binary trees:

```
class Node
{
public:
    int key_;
    Node * parent_;
    Node * left_;
    Node * right_;
};
```

Write a function `deallocate` such that if `proot` points to the root of a binary tree (with nodes in the heap), `deallocate(proot)` will deallocate the memory used by this tree and set `proot` to `NULL`.

Answer:

Q18. Describe an algorithm f such that $f(x, n, k)$ would compute the k–th smallest value in an array $x[0..n-1]$ of doubles. For instance if the array is

$$x = \{1.1, 5.5, 2.2, 7.7, 3.3, 4.4\}$$

then $f(x, n, 3)$ would return the third smallest value of x, which is 3.3. Note that you want the fastest algorithm.

(Hint: Modify quicksort.)

Answer: