University of Waterloo
Department of Electrical and Computer Engineering
ECE250
Algorithms and Data Structures
Fall 2014

Midterm Examination

Instructor: Ladan Tahvildari, PhD, PEng, SMIEEE
Date: Tuesday, October 21, 2014, 10:00 a.m.
Location: EIT-1015; CPH-3602; CPH-3604
Duration: 80 minutes
Type: Closed Book

Instructions:

- There are 4 questions. Answer all 4 questions.
- Standard calculator allowed but no additional materials allowed.
- The number in brackets denotes the relative weight of the question (out of 100).
- If information appears to be missing from a question, make a reasonable assumption, state it and proceed.
- Write your answers directly on the sheets.
- If the space to answer a question is not sufficient, use overflow page.
- When presenting programs, you may use any mixture of pseudocode/C++ constructs as long as the meaning is clear.

Name	Student ID

Question	Mark	Max	Marker
1	A: B: C:	35	A: B: C:
2		10	
3	A: B: C:	35	A: B: C:
4	A: B:	20	A: B:
Total		100	

Name: **Student ID:**

Question 1: Algorithm Analysis [35]

Part A [15].

Consider the following recursive algorithm that computes minimum value out of real numbers stored in the array A on positions from l to r.

FindMin(A: array of real numbers, l: integer, r: integer)

> ***if*** $l = r$ ***then return*** *A[l]*
> $temp_1 \leftarrow FindMin(A, l, \left\lfloor \frac{l+r}{2} \right\rfloor)$
> $temp_2 \leftarrow FindMin(A, \left\lfloor \frac{l+r}{2} + 1 \right\rfloor, r)$
> ***if*** $temp_1 \prec temp_2$
> ***then return*** $temp_1$
> ***else return*** $temp_2$

Write a recurrence describing the cost of running this algorithm on n element array: *FindMin(A,1,n)*. Solve the recurrence and give the resulting running time of the algorithm.

The algorithm divides the problem into two parts; each one roughly half of the size of the original problem. Then, it combines the two parts using a constant time operation.

$$\begin{cases} T(1) = O(1) \\ T(n) = T\left\lfloor \frac{n}{2} \right\rfloor + T\left\lceil \frac{n}{2} \right\rceil + O(1) \end{cases}$$

$$T(n) = 2T(n/2) + O(1)$$

$$\left.\begin{array}{r} a = 2, b = 2, f(n) = O(1) \\ n^{\log_b a} = n^1 = n \end{array}\right\} \rightarrow f(n) = O\left(n^{\log_b a - \varepsilon}\right) \text{ for } \varepsilon = 1 \quad \textit{(Case 1)}$$

$$T(n) = \Theta\left(n^{\log_b a}\right) = \Theta(n)$$

FindMin algorithm runs in linear time.

Part B. [10]

Give asymptotic upper and lower bounds for the following recurrence. Assume that $T(n)$ is constant for sufficiently small n. Make your bounds as tight as possible, and justify your answer.

$$T(n) = 16T(n/4) + n^5\sqrt{n}$$

a = 16, b = 4 $\rightarrow n^{\log_b a} = n^{\log_4 16} = n^2$

f(n) = $n^5\sqrt{n} = n^{\frac{11}{2}} = \Omega(n^{2+\varepsilon})$; $\varepsilon = \frac{7}{2} = 3.5$

This is Case 3 of Master Method

Checking the Regularity Condition

af(n/b) ≤ cf(n)

$\rightarrow 16\left(\frac{n}{4}\right)^5 * \sqrt{\frac{n}{4}} \leq cn^5\sqrt{n}$

$\rightarrow \frac{16n^5}{4^5} * \frac{\sqrt{n}}{2} \leq cn^5\sqrt{n}$

$\rightarrow \frac{1}{128} \leq c$

for some $\frac{1}{128} \leq c < 1$, *af(n/b) ≤ cf(n)*

Therefore, Case 3 applied $\rightarrow T(n) = \Theta(n^5\sqrt{n})$

Name: **Student ID:**

Part C. [10]

Prove that $f(n) = 10^7 + 7n^7 \log n + 3n^7$ is $O(n^7 \log n)$ using the definition of "Big-Oh".

Big-Oh Definition:

$$f(n) \le cg(n), \quad \text{there exist} \quad c \quad \& \quad n_0 : \quad n \ge n_0$$

$$10^7 + 7n^7 \log n + 3n^7 \le \quad cn^7 \log n$$

$$10^7 + 7n^7 \log n + 3n^7 \le 10^7 n^7 \log n + 7n^7 \log n + 3n^7 \log n$$

$$\downarrow$$

$$(10^7 + 7 + 3)n^7 \log n \le cn^7 \log n$$

$c = 10^7 + 10$ and $n \ge 2$; $n_0 = 2$

Name: **Student ID:**

Question 2: Elementary Data Structures [10]

You have been provided with an implementation of a *stack-of-integers* ADT. This implementation includes the usual operations such as ***Push***, ***Pop***, and ***IsEmpty***.

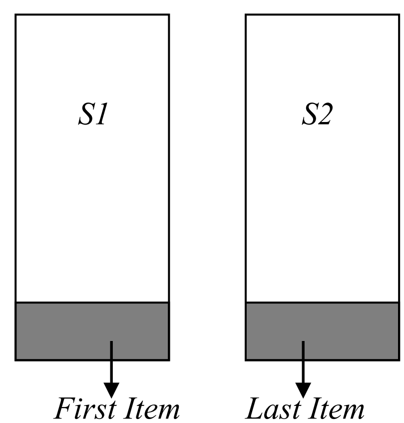
Describe how you would implement the Queue ADT using two stacks. Specifically, give algorithms for the ***Enqueue*** and ***Dequeue*** operations. Give tight Big-Oh expressions for the running times of your implementation.

You may assume that ***Push***, ***Pop***, and ***IsEmpty*** are all $O(1)$.

Several different implementations are possible. Here is one:

```
class Queue
{
        stack s1;
        stack s2;
public:
        void Enqueue(int);
        int Dequeue();
};

void Queue::Enqueue(int item)
{
        while (!s2.IsEmpty())
                s1.Push(s2.Pop());
        s1.Push(item);
}
```



The running time of Enqueue is O(n).

```
void Queue::Dequeue()
{
        while (!s1.IsEmpty())
                s2.Push(s1.Pop());
        return s2.Pop;
}
```

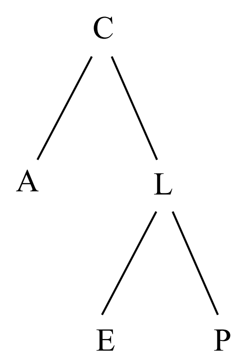
The running time of Dequeue is O(n).

Name: **Student ID:**

Question 3: Trees and Tree Traversals [35]

Part A. [15]

- Consider we have a BST contains the following keys: P, L, A, C, E.
  Draw this BST in such a way that the result of the "preorder tree traversal" is: C, A, L, E, P.



- Let T be an AVL tree of height 6. What is the smallest number of nodes it can store?

  Let N_h be the smallest number of nodes for an AVL tree of height h.

  In class, we have shown that:

$$N_0 = 1 \;;\; N_1 = 2 \;;\; N_h = N_{h-2} + N_{h-1} + 1$$

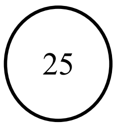
$$\begin{aligned} N_6 &= N_4 + N_5 + 1 \\ &= N_4 + (N_3 + N_4 + 1) + 1 \\ &= (N_2 + N_3 + 1) + N_3 + (N_2 + N_3 + 1) + 2 \\ &= 2N_2 + 3N_3 + 4 \\ &= 5N_2 + 3N_1 + 7 \\ &= 5N_0 + 8N_1 + 12 \\ &= 5 + 16 + 12 \\ &= 33 \end{aligned}$$

Name: **Student ID:**

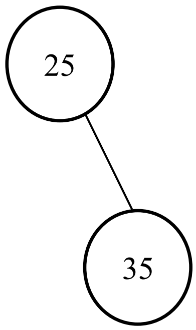
Part B. [12]

Start with an empty AVL tree and insert the following keys in the given order: 25, 35, 45, 20, 22, and 27. Draw the trees following each insertion, and also after each rotation. Specify the rotation types.

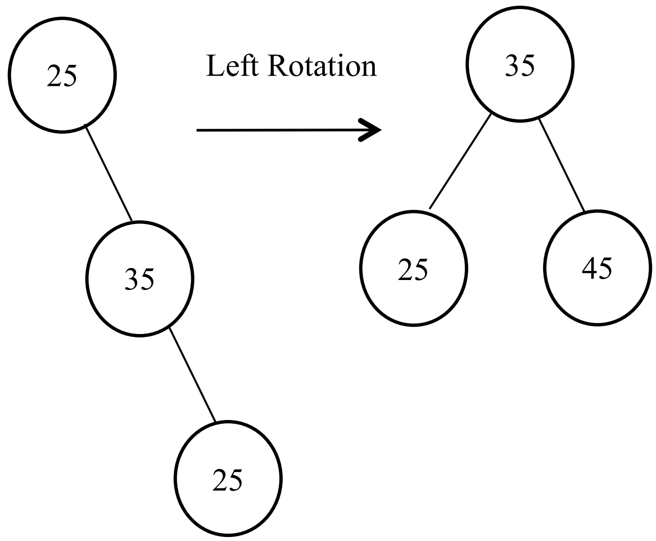
Insert 25



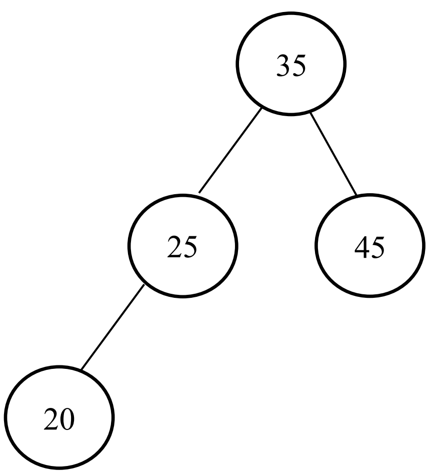
Insert 35



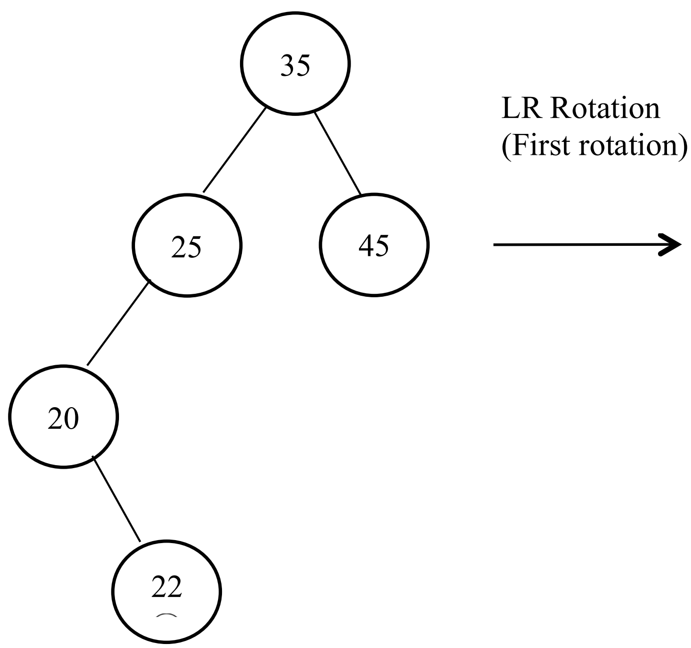
Insert 45



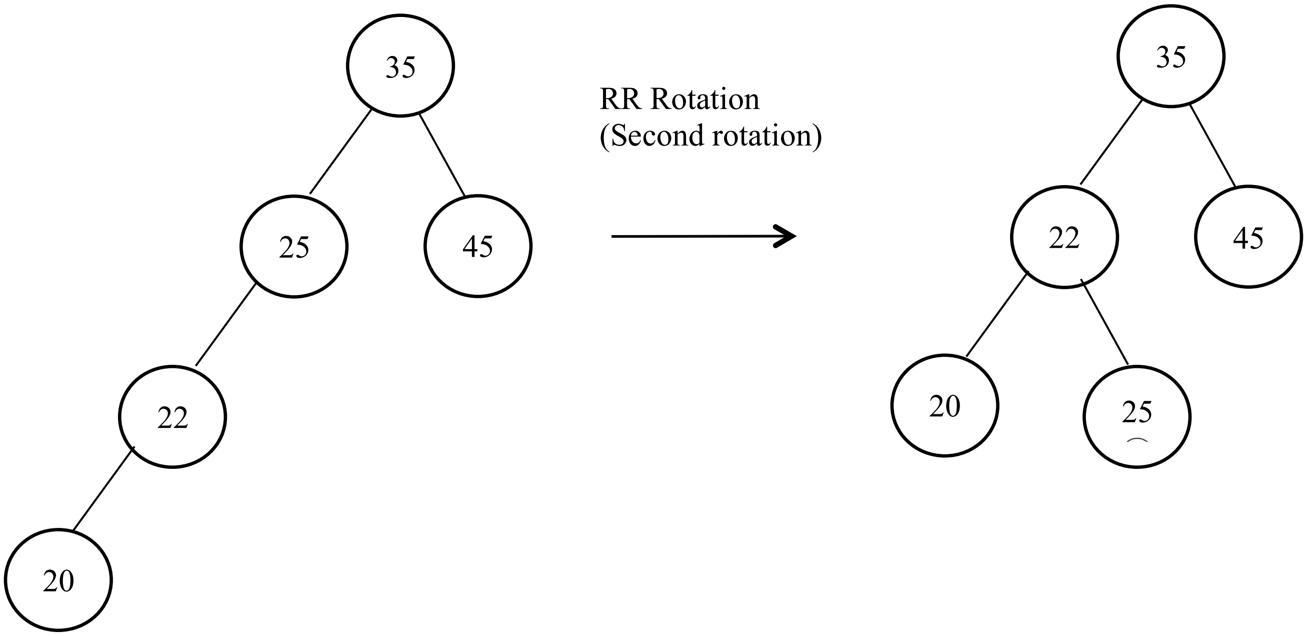
Insert 20

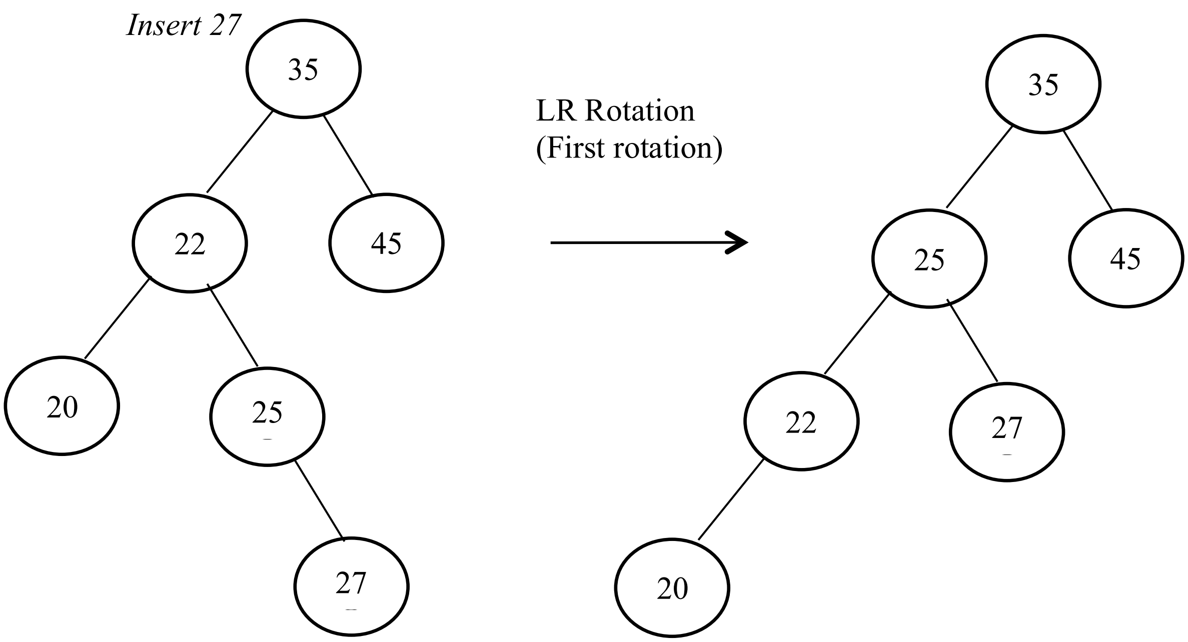


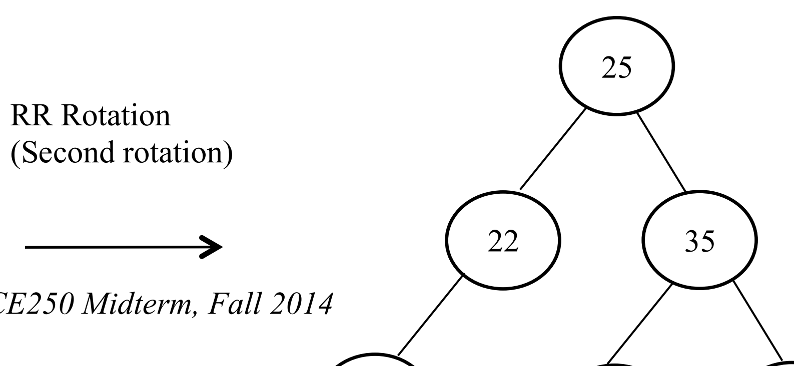
Insert 22



Name: **Student ID:**



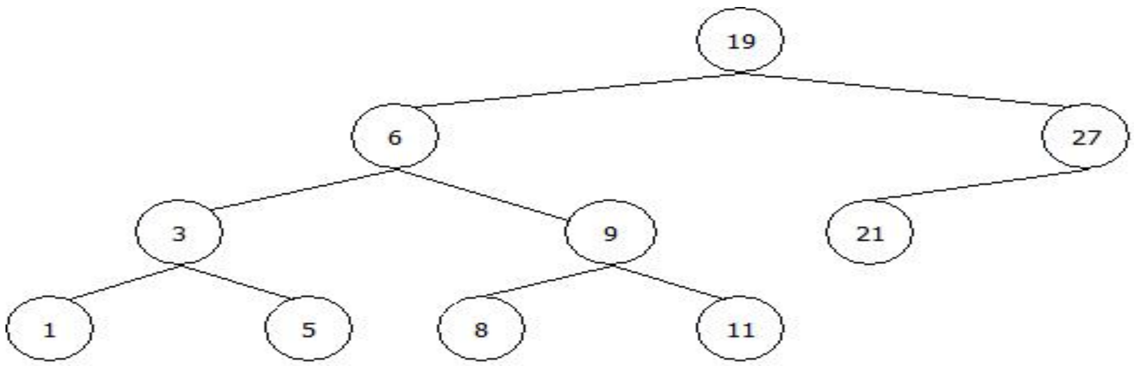


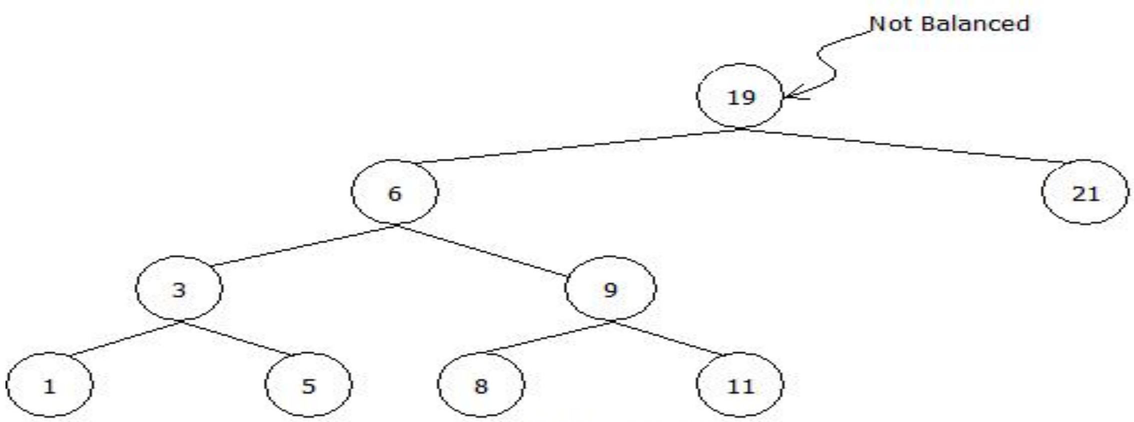


Name: **Student ID:**

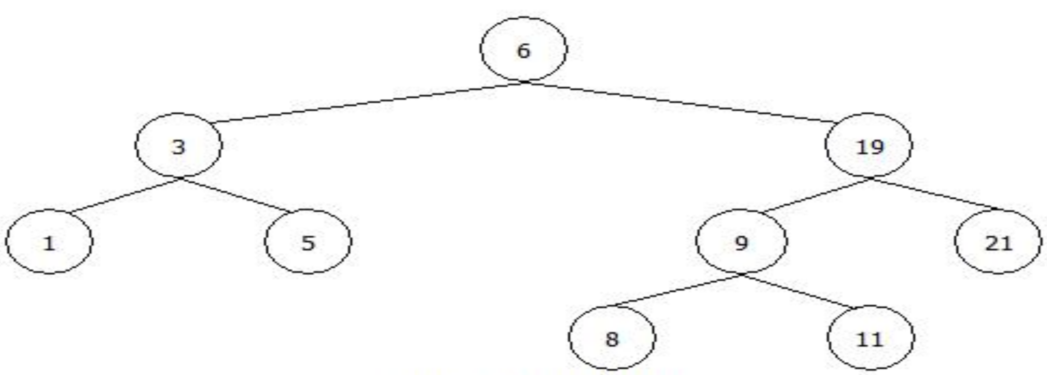
Part C. [8]

Assume we have the following AVL tree. Show the tree after deleting 27. Use the successor of a node if it is needed. Draw the final tree and all intermediate trees.





Deletion Key 27



Just one (Right Rotation)

Name: **Student ID:**

Question 4: Hashing [20]

Part A. [5]

Suppose we have a hash table with N slots containing n keys. Suppose that instead of a linked list, each slot is implemented as a binary search tree. Give the worst and the best time complexity of adding an entry to this hash table. Explain your answers.

- *Best case happens when we insert into an empty slot [1] → this is $O(1)$ time.*
- *Worst case happens when we insert in a slot that has all "n" keys which equals to "worst case" insertion into BST (sorted list) which means $O(n)$.*

Name: **Student ID:**

Part B. [15]

Consider a hash table of size 7. Suppose the hash function uses division method. Insert, in the given order, keys: 2, 4, 12, 19, and 20 into the hash table using:

- Linear probing to resolve the collisions. Show all your work.

  $h_1(k) = k \; mod \; 7$
  $h(k, i) = \; (h_1(k) + i) \; mod \; 7$
  $h_1(2) = 2 \; mod \; 7 = 2$

  $h_1(4) = 4 \; mod \; 7 = 4$

  $h_1(12) = 12 \; mod \; 7 = 5$

  $h_1(19) = 19 \; mod \; 7 = 5 \rightarrow$ Collision

  $h(19,1) = (h_1(19) + 1) \; mod \; 7 = 6$

  $h_1(20) = 20 \; mod \; 7 = 6 \rightarrow$ Collision

  $h(20,1) = (h_1(20) + 1) \; mod \; 7 =$
  $7 \; mod \; 7 = 0$

0	20
1	
2	2
3	
4	4
5	12
6	19

- Double hashing to resolve collisions with the secondary hash function $h_2(k) = 5 - \left(k \quad \text{mod} \quad 5 \right)$. Show all your work.

  $h_1(k) = k \; mod \; 7$
  $h(k, i) = \; (h_1(k) + i \, . \, h_2(k)) \; mod \; 7$
  $h_1(2) = 2 \; mod \; 7 = 2$
  $h_1(4) = 4 \; mod \; 7 = 4$
  $h_1(12) = 12 \; mod \; 7 = 5$
  $h_1(19) = 19 \; mod \; 7 = 5 \rightarrow$ Collision
  $h(19,1) = (h_1(19) + 1 \cdot h_2(19)) \; mod \; 7$
  $= (5 + 1 \cdot 1) \; mod \; 7 = 6$
  $h_1(20) = 20 \; mod \; 7 = 6 \rightarrow$ Collision
  $h(20,1) = (h_1(20) + 1 \cdot h_2(20)) \; mod \; 7 = (6 + 1 \cdot$
  $5) \; mod \; 7 = 4 \rightarrow$ Collision
  $h(20,2) = (h_1(20) + 2 \cdot h_2(20)) \; mod \; 7 = (6 + 2 \cdot$
  $5) \; mod \; 7 = 2 \rightarrow$ Collision
  $h(20,3) = (h_1(20) + 3 \cdot h_2(20)) \; mod \; 7 = (6 + 3 \cdot$
  $5) \; mod \; 7 = 0$

0	20
1	
2	2
3	
4	4
5	12
6	19

Name: **Student ID:**

OVERFLOW SHEET [Identify the question(s) being answered.]