CISC-235*
Winter 2016
Test 2
February 25, 2016

Student Number (Required) ______________________

Name (Optional) ________________________________

This is a closed book test. You may not refer to any resources.

This is a 50 minute test.

Please write your answers in ink. Pencil answers will be marked, but will not be re-marked under any circumstances.

The test will be marked out of 50.

<table>
<thead>
<tr>
<th>Question 1</th>
<th>/12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 2</td>
<td>/12</td>
</tr>
<tr>
<td>Question 3</td>
<td>/12</td>
</tr>
<tr>
<td>Question 4</td>
<td>/2</td>
</tr>
<tr>
<td>Question 5</td>
<td>/12</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>/50</td>
</tr>
</tbody>
</table>
Question 1 (12 marks)

In Red-Black trees, one of the colouring rules is that a Red vertex cannot have a Red child. What is the purpose of this rule?
Question 2 (12 marks):

The last step of the insertion algorithm for Red-Black Trees is “Colour the root Black”.

a) [6 marks] Explain why the colour of the root vertex of a Red-Black tree is actually irrelevant for ensuring that the “black-height” balance is maintained.

b) [6 marks] Explain why this step is included in the algorithm.
Question 3 (12 marks):

a) [6 marks] Prove that if the root of a Red-Black tree has two Red children, we could colour them both Black without affecting the validity of the colouring.

b) [6 marks] Show that if the root of a Red-Black tree has two Black children, it is not necessarily true that we can colour them both Red without affecting the validity of the colouring.
Question 4 (2 marks):

If a Red-Black tree is constructed using the insertion algorithm we have studied, and contains at least 2 values, then it always contains at least one Red vertex.

Choose one answer:

True

False
Question 4 (12 marks):

Here is an insert function for a hash table:

```c
insert(k):
    a = h(k)
    i = 0
    while i < m:
        p = (a + 2*i) % m
        if T[p] == 'empty' or T[p] == 'deleted':
            T[p] = k
            return "success"
        else:
            i += 1
    return "fail"
```

This is simply linear probing using a step size of 2 instead of 1. You may assume that m (the size of the hash table) is odd.

Does this reduce primary clustering, when compared to standard linear probing? (Hint: what do the probe sequences look like?)
Student Number: ________________

Nearly Blank Page