# King Fahd University of Petroleum & Minerals

## College of Computer Science and Engineering

Information and Computer Science Department
First Semester 211 (2021/2022)

ICS 202 – Data Structures and Algorithms
Final Exam
Saturday, 25th December 2021
Time: 120 minutes

Name: ______________________________________________

| ID# | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|

Section: ____________

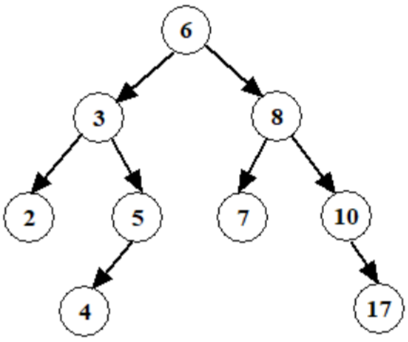
1. This exam consists of 12 pages including the title page.
2. Calculators are not allowed.
3. Mobile phones are not allowed

| Question # | Max Marks | Marks Scored | Comments |
|---|---|---|---|
| 1 [AVL Trees/Heaps] | 20 | | |
| 2 [B and B+ Trees] | 20 | | |
| 3 [Graphs and Graph Algorithms] | 25 | | |
| 4 [Hashing] | 15 | | |
| 5 [Data Compression/String Matching] | 20 | | |
| Total | 100 | | |

Q. 1 [AVL Trees and Binary Heaps]

(a) [5 + 5 = 10 marks] Draw the **final** AVL tree after each of the following operations (**you need to draw two AVL trees**), and mention the type of rotations,

i. when the key **12,** and then
ii. the key **20** are inserted in the following AVL tree.



<table>
<tr>
<td>Final AVL Tree after 12 is inserted in the above AVL Tree

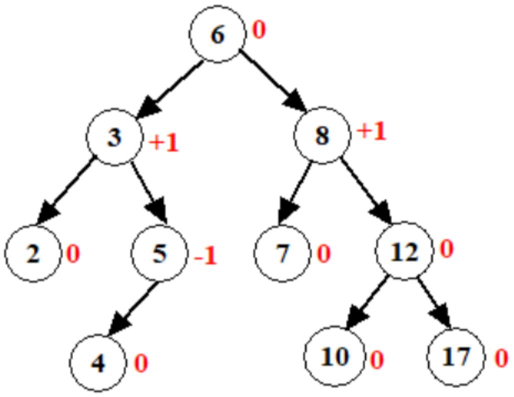
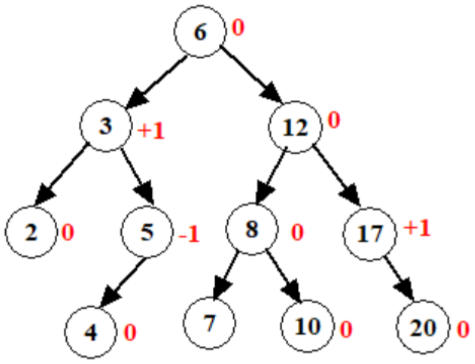


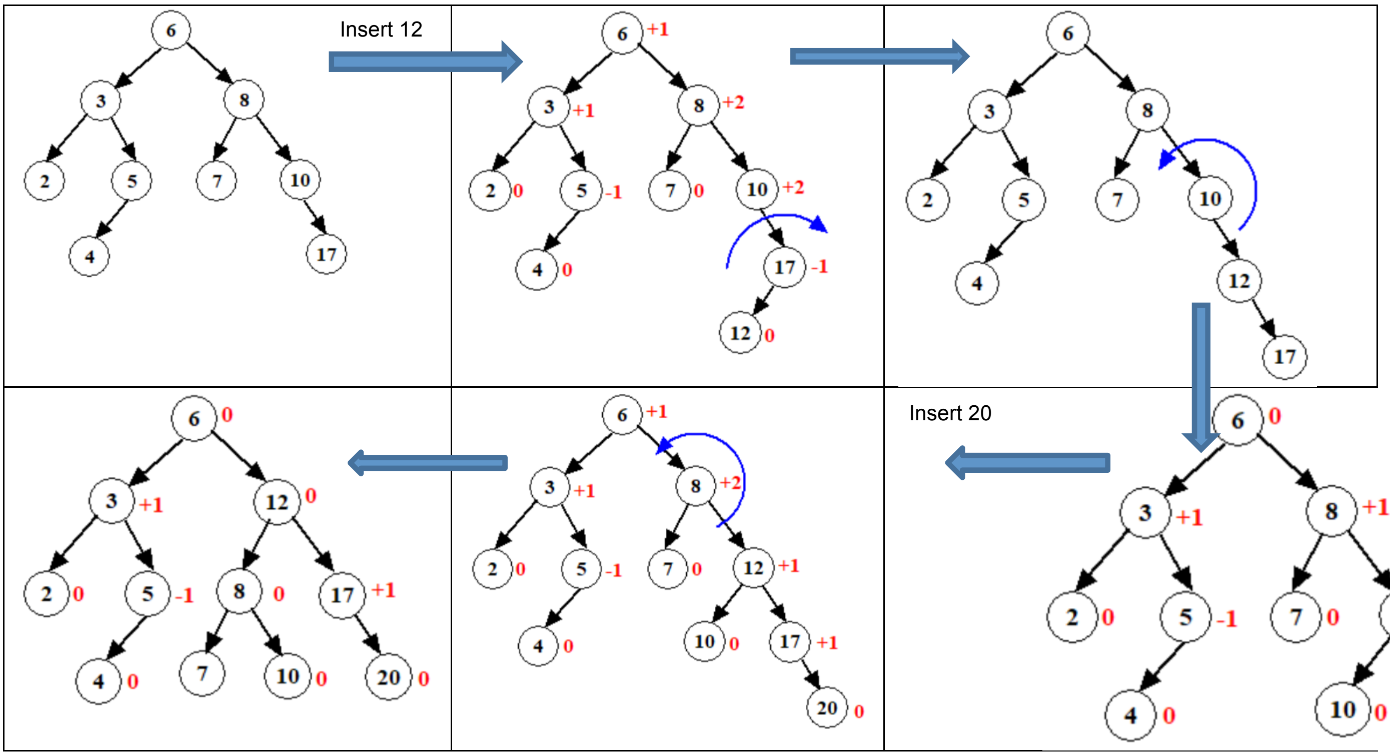
Figure 1</td>
<td>Final AVL Tree after 20 is inserted in the tree on the **left (i.e. in Figure 1)**


</td>
</tr>
<tr>
<td>Type of Rotation:

Double Right Left</td>
<td>Type of Rotation:

Single Left</td>
</tr>
</table>

Note: Students are not required to draw all trees



(b) [2 marks] For a MaxHeap in which indexes start at 1.

- What is the index of the parent of a node with index **k**? ____k/2____
- What is the index of the right child, if any, of a node with index **k**? _2k + 1______

(c) [4 marks] Show the result of converting the following array into a min-heap using the bottom-up approach:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **65** | **24** | **63** | **60** | **27** | **14** | **11** | **8** | **21** | **26** |

Resulting min-heap:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **8** | **21** | **11** | **24** | **26** | **14** | **63** | **60** | **65** | **27** |

(d) [4 marks] Draw the heap, as an array, after deleting the maximum from the following max-heap:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 20 | 18 | 14 | 12 | 11 | 13 | 7 | 6 | 8 | 5 | 9 | 4 |

Resulting max-heap:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **18** | **12** | **14** | **8** | **11** | **13** | **7** | **6** | **4** | **5** | **9** | |

Q. 2 [B Trees and B+ Trees]

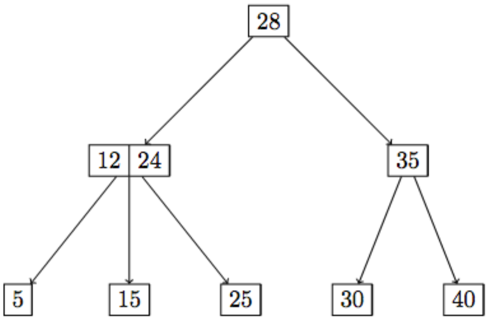
(a) [2 marks] For a non-empty B-Tree of order m, fill in the following table :

| B-Tree | Root Node | Non-root Node |
|---|---|---|
| Maximum number of keys | m - 1 | m – 1 |
| Maximum number of non-empty subtrees | m | m |
| Minimum number of keys | 1 | ⌈m/2⌉ – 1 |
| Minimum number of non-empty subtrees | 2 | ⌈m/2⌉ |

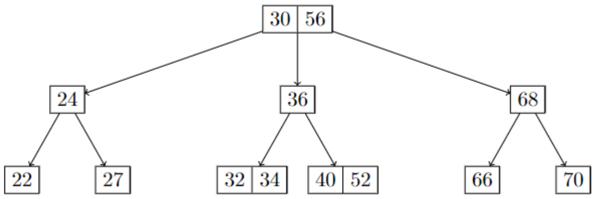
(b) [6 marks] Draw the **final** B-tree when the following keys are inserted, in the given order, into an empty B-Tree of order 3.

24, 35, 28, 30, 25, 12, 15, 40, 5

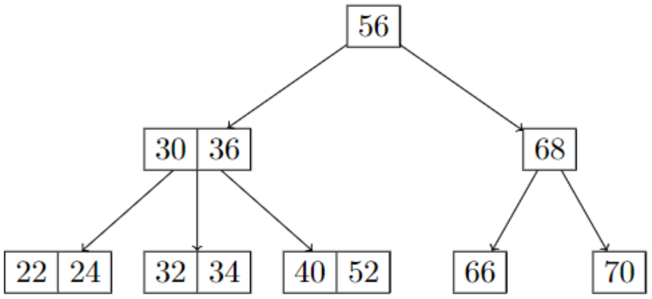
Answer:



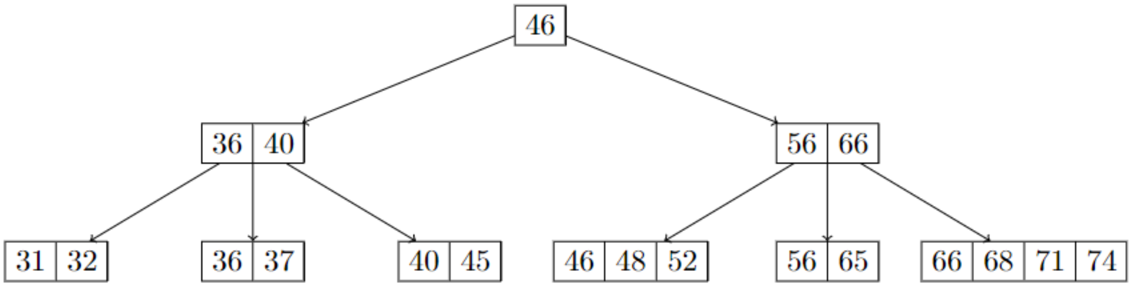
(c) [6 marks] Draw the **final** B-tree when the key **27** is deleted from the following B-tree of order 3:



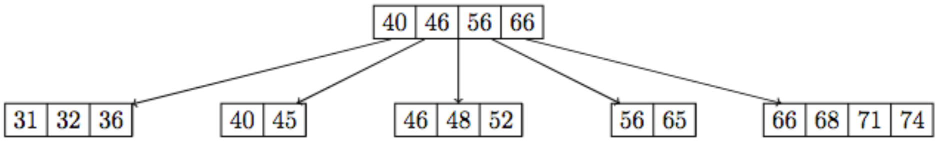
Answer:



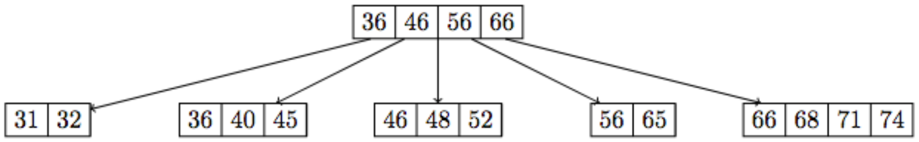
(d) [6 marks] Draw the **final** B+ tree when the key **37** is deleted from the following B+ tree of order M = 5 and L = 4



Answer:

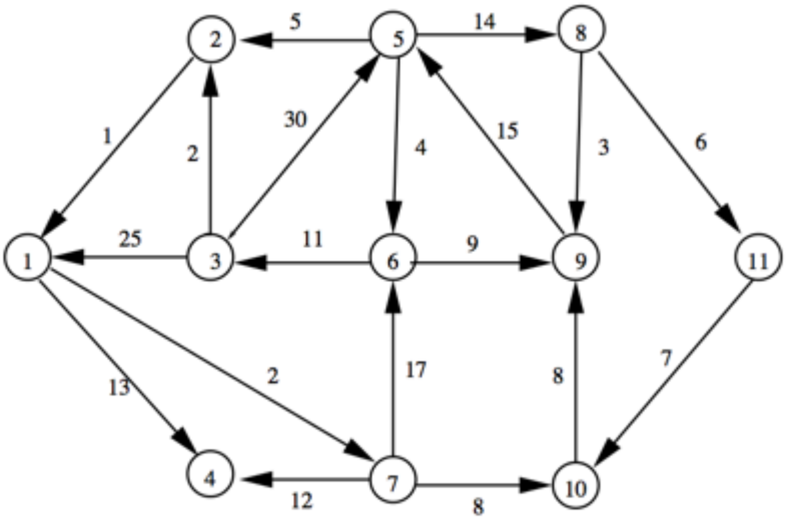


OR



Q. 3: **[25 points] (Graphs):**

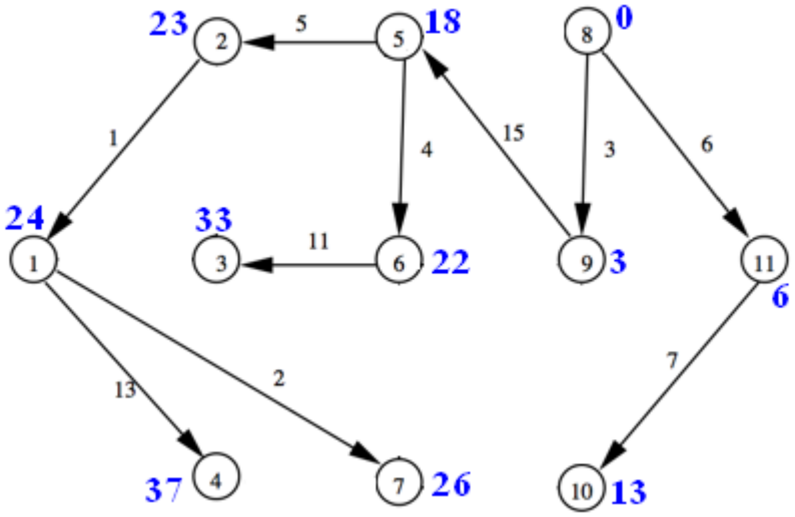
A. [6 marks] Consider the following weighted directed graph G:



Apply the Dijkstra algorithm on G starting from **vertex 8**:

| | Initially | | | | | | | | | | | | Weight | Pred |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| active | | 8 | 9 | 11 | 10 | 5 | 6 | 2 | 1 | 7 | 3 | 4 | | |
| 1 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | 24 | # | # | # | # | 24 | 2 |
| 2 | ∞ | ∞ | ∞ | ∞ | ∞ | 23 | 23 | # | # | # | # | # | 23 | 5 |
| 3 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | 33 | 33 | 33 | 33 | # | # | 33 | 6 |
| 4 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | 37 | 37 | 37 | # | 37 | 1 |
| 5 | ∞ | ∞ | 18 | 18 | 18 | # | # | # | # | # | # | # | 18 | 9 |
| 6 | ∞ | ∞ | ∞ | ∞ | ∞ | 22 | # | # | # | # | # | # | 22 | 5 |
| 7 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | 26 | # | # | # | 26 | 1 |
| 8 | 0 | # | # | # | # | # | # | # | # | # | # | # | 0 | - |
| 9 | ∞ | 3 | # | # | # | # | # | # | # | # | # | # | 3 | 8 |
| 10 | ∞ | ∞ | ∞ | 13 | # | # | # | # | # | # | # | # | 13 | 11 |
| 11 | ∞ | 6 | 6 | # | # | # | # | # | # | # | # | # | 6 | 8 |

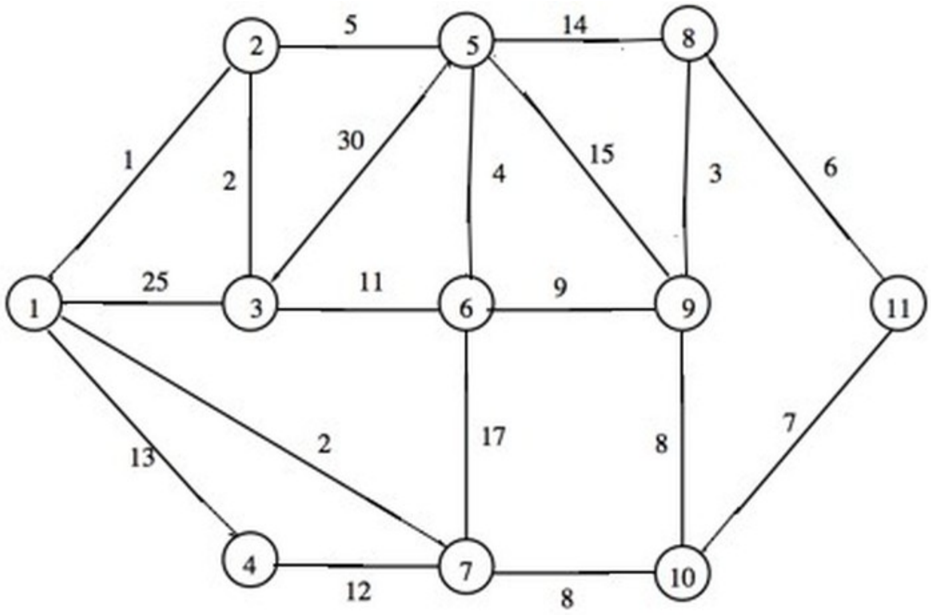
B. [4 marks] Draw the shortest path tree.



C. [5 marks] Fill in the following table with the big-O complexity of each operation

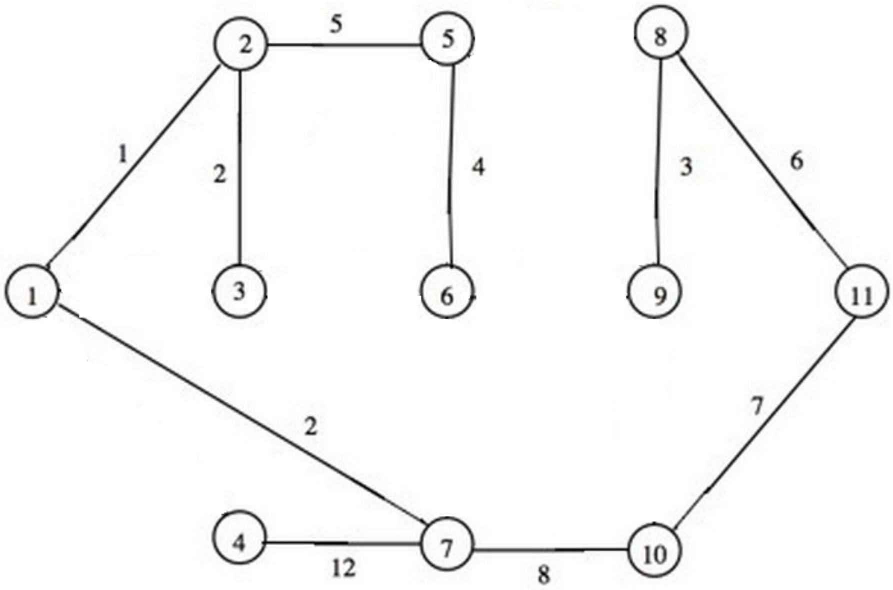
| **Operation / Data Structure** | **Adjacency Matrix** | **Adjacency List** |
|---|---|---|
| Is there an edge from **x** to **y** | O(1) | O(n) |
| Edge Insertion | O(1) | O(n) |
| Edge deletion | O(1) | O(n) |
| Visit all edges | O($n^2$) | O(n+m) |
| Space complexity | O($n^2$) | O(n+m) |

D. [6 marks] Use Prim’s algorithm to find a minimum spanning tree of the graph below starting from Vertex 8.



| active | Initially | 8 | 9 | 11 | 10 | 7 | 1 | 2 | 3 | 5 | 6 | 4 | Weight | V1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ∞ | ∞ | ∞ | ∞ | ∞ | 2 | # | # | # | # | # | # | 2 | 7 |
| 2 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | 1 | # | # | # | # | # | 1 | 1 |
| 3 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | 25 | 2 | # | # | # | # | 2 | 2 |
| 4 | ∞ | ∞ | ∞ | ∞ | ∞ | 12 | 12 | 12 | 12 | 12 | 12 | # | 12 | 7 |
| 5 | ∞ | 14 | 14 | 14 | 14 | 14 | 14 | 5 | 5 | # | # | # | 5 | 2 |
| 6 | ∞ | ∞ | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 4 | # | # | 4 | 5 |
| 7 | ∞ | ∞ | ∞ | ∞ | 8 | # | # | # | # | # | # | # | 8 | 10 |
| 8 | 0 | # | # | # | # | # | # | # | # | # | # | # | 0 | - |
| 9 | ∞ | 3 | # | # | # | # | # | # | # | # | # | # | 3 | 8 |
| 10 | ∞ | ∞ | 8 | 7 | # | # | # | # | # | # | # | # | 7 | 11 |
| 11 | ∞ | 6 | 6 | # | # | # | # | # | # | # | # | # | 6 | 8 |

E. [4 marks] Draw the resulting minimum spanning tree.



Q. 4: [5+5+5=15 marks] Given a hash table with the size 7 and hash function **h(key) = key % 7,** draw the final state of the hash-table with the following collision resolution techniques:

insert (14), insert(2), insert (42), insert (8), insert (21)

(a) open addressing with $c(i) = \pm i^2$

| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| Key | **14** | **42** | **2** | | | **8** | **21** |

| |
|---|
| (1) $h_0(14) = 14 \% 7 = \mathbf{0}$ |
| (2) $h_0(2) = 2 \% 7 = \mathbf{2}$ |
| (3) $h_0(42) = 42 \% 7 = 0$ [collision], $h_1(42) = (0 + 1^2) \% 7 = \mathbf{1}$ |
| (4) $h_0(8) = 8 \% 7 = 1$ [collision], $h_1(8) = (1 + 1^2) \% 7 = 2$ [collision], |
| $h_{-1}(8) = (1 - 1^2) \% 7 = 0$ [collision], $h_2(8) = (1 + 2^2) \% 7 = \mathbf{5}$ |
| (5) $h_0(21) = 21 \% 7 = 0$ [collision], $h_1(21) = (0 + 1^2) \% 7 = 1$ [collision], |
| $h_{-1}(21) = (0 - 1^2) \bmod 7 = -1 \bmod 7 = (-1+7) \bmod 7 = \mathbf{6}$ |

(b) open addressing with $c(i) = i * h_p(x)$, where $h_p(x) = 1 + x \bmod 6$.

| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| Key | **14** | **42** | **2** | | **8** | **21** | |

| |
|---|
| (1) $h_0(14) = 14 \% 7 = \mathbf{0}$ |
| (2) $h_0(2) = 2 \% 7 = \mathbf{2}$ |
| (3) $h_0(42) = 42 \% 7 = 0$ [collision], $h_p(42) = (1 + 42 \% 6) = 1$, $h_1(42) = (0 + 1*1) \% 7 = \mathbf{1}$ |
| (4) $h_0(8) = 8 \% 7 = 1$ [collision], $h_p(8) = (1 + 8 \% 6) = 3$, $h_1(8) = (1 + 1*3) \% 7 = \mathbf{4}$ |
| (5) $h_0(21) = 21 \% 7 = 0$ [collision], $h_p(21) = (1 + 21 \% 6) = 4$, |
| $h_1(21) = (0 + 1 * 4) \% 7 = 4$ [collision], $h_2(21) = (0 + 2*4) \% 7 = \mathbf{1}$ [collision], |
| $h_3(21) = (0 + 3*4) \% 7 = \mathbf{5}$ |

(c) separate chaining (using above computations)

| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| Key | **14** | **8** | **2** | | | | |

↓

**42**

↓

**21**

| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| Key | **14** | **8** | **2** | | | | |

↓

**21**

↓

**42**

Q.5: [20 marks]

5 (a): (i) [8 marks] Compress the string: WEWEDWEEDWEB using the **LZ78** compression algorithm. Show all details of your work using a properly labeled table (i.e. you **must** indicate the contents of each column in your table)

| Output | Dictionary Index | Dictionary String |
|---|---|---|
| **(0, W)** | **1** | **W** |

|  |  |  |
|---|---|---|
| **(0, E)** | **2** | **E** |
| **(1, E)** | **3** | **WE** |
| **(0, D)** | **4** | **D** |
| **(3, E)** | **5** | **WEE** |
| **(4, W)** | **6** | **DW** |
| **(2, B)** | **7** | **EB** |

The compressed string is (write your answer here):

**0W 0E 1E 0D 3E 4W 2B**

Q. 5 (a) (ii) [2 marks] Calculate the compression ratio.

Bits in the original string
= 12 * 8 = 96

Bits in the compressed string
= (1 + 8) + (1 + 8) + (2 + 8) + (2 + 8) + (3 + 8) + (3 + 8) + (3 + 8) = 71 bits
**0W 0E 1E 0D 3E 4W 2B**

Compression Ratio = 96/71 = 1.35

The compression ratio is
(write your answer here)

**96 / 71 = 1.352**

Q. 5 (b) (i) [8 marks] Find the **next** array for the Knuth Morris Pratt Algorithm for the pattern “ONIONS”. Fill in the following table to find the **next** array.

| *j* | Pattern [0..*j* - 1] | Proper Prefixes | Proper Suffixes | Length of border of P[0..*j* – 1] | next[*j*] |
|---|---|---|---|---|---|
| 0 | - | null | null | –1 (defined) | –1 |
| 1 | O | - | - | **0** | **0** |
| 2 | ON | O | N | **0** | **0** |
| 3 | ONI | O, ON | I, NI | **0** | **0** |
| 4 | ONIO | O, ON, ONI | O, IO, NIO | **1** | **1** |
| 5 | ONION | O, ON, ONI, ONIO | N, ON, ION, NION | **2** | **2** |
| 6 | ONIONS | O, ON, ONI, ONIO, ONION | S, NS, ONS, IONS, NIONS | **0** | **0** |

Therefore, the next array is (write your answer here):

| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| next[index] | -1 | **0** | **0** | **0** | **1** | **2** | **0** |

**Figure 2**

Q. 5 (b) (ii) [2 marks] Consider the following text = “ONIONIONSPL” and the pattern = “ONIONS”. Given the initial matching configuration where text[0] is matched with pattern[0], and a mismatch occurs at the last character of the pattern i.e., a mismatch at index *i* = 5 of the text and *j* = 5 of the pattern as shown,

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Text | O | N | I | O | N | I | O | N | S | P | L |
| Pattern | O | N | I | O | N | S | | | | | |

Using the Knuth-Morris-Pratt Algorithm (and the next array as in Figure 2), what are the values of *i* for the text and *j* for the pattern at which the search will resume? (write your answer here).

Answer: *i* = **5** and *j* = **2**