The University of Melbourne

Department of Computing and Information Systems

COMP90049
Knowledge Technologies
Sample Solutions
Mid-semester Test

Length: This paper has 3 pages including this cover page.

Authorised materials: None

Calculators: Not permitted

Time: 50 minutes, with no reading time

Instructions to students: This exam is worth a total of 10 marks and counts for 10% of your final grade. Please answer all questions in the provided spaces on the test page. Please write your student ID in the space provided below. The test may not be removed from the test venue.

Student id:

Examiner's use only:

Q1	*Q2*	*Q3*	*Q4*	*Q5*

Continued overleaf . . .

COMP90049 Knowledge Technologies
Mid-semester Test

Semester 1, 2013

Total marks: 10

Students must attempt all questions

1. Describe, with the aid of an example, the difference between "concrete tasks" and "knowledge tasks." (1 mark)

   - Concrete tasks: task is well-defined, "correct" solution, data is transformed in a mechanical way; no context for users, limited contribution to human knowledge; e.g. add two numbers, crack a code
   - Knowledge tasks: data is unreliable, outcome is not well-defined, no single "correct" solution (typically many "good" solutions); computers mediate between users and data, context (for the user) is critical, enhance human knowledge; e.g. relevant results for a web query, translation between languages

2. For the "regular expression":

   ```
   \S(he)a[t]{1,2}i?
   ```

   which of the following strings would the expression match (circle each)? (1 mark)

   (a) `eai`
   (b) `heatt`
   (c) `cheaters`
   (d) `space heating`

   - (c)

3. Consider searching for the substring `gone` within the target string `wegoandthenwearegoners` — indicate the behaviour of the Boyer-Moore algorithm (as discussed in this subject), including any necessary data structures. (3 marks)

   - Data structure is a lookup table: `g` → 3, `o` → 2, `n` → 1, `e` → 0 (match backward), otherwise → 4
   - Start at position 4, read `o`, look up `o` in table, read 2, jump to position 6
   - Read `n`, look up `n` in table, read 1, jump to position 7
   - Read `d`, look up `d` in table, read 4, jump to position 11
   - Read `n`, look up `n` in table, read 1, jump to position 12
   - Read `w`, look up `w` in table, read 4, jump to position 16
   - Read `e`, look up `e` in table, read 0, attempt to match backwards, find mismatch (with `r`), jump to position 20
   - Read `e`, look up `e` in table, read 0, attempt to match backwards, match `n`, match `o`, match `g` — match found

Continued overleaf . . .

4. Use the "global edit distance," as shown in the lectures, to find the distance **from** the string `lede` **to** the string `deed`, based on the following parameter vector:

   $[m, i, d, r] = [-3, 1, 4, 2]$

   Use as much of the matrix below as you need. There are four paths through the matrix corresponding to the optimal distance; write the operations for at least two of them to the right of the table. (3 marks)

	ε	l	e	d	e
ε	0 ← ↑ ↖	4 ← ↖	8 ← ↖	12 ←	16
d	1 ↑ ↖	2 ← ↑ ↖	6	5 ← ↖	9
e	2 ↑ ↖	3 ↑ ↖	-1 ← ↑ ↖	3 ↖	2
e	3 ↑ ↖	4 ↑	0 ↑ ↖	1	0 ↑
d	4	5	1	-3 ←	1

   - Edit distance: 1; Operations: `rmdmi` or `rmimd` or `rimmd` or `irmmd`

5. For the "Soundex algorithm":

   (a) Apply it to the strings `carter` and `clinton`, using the following modified table: (1 mark)

`aeiouwy`	0
`bdgjlmnrvz`	1
`cfhkpqstx`	2

   - `carter`:
     - `c01201`
     - `c01201`
     - `c121`
     - `c121`
   - `clinton`
     - `c101201`
     - `c101201`
     - `c1121`
     - `c112`

   (b) Briefly describe how you might use your results in part (a) to perform "approximate matching" for the string `collins`. (1 mark)

   - Find the Soundex code for `collins`
   - Find the (global) edit distance (or 2-gram distance, or...) between the Soundex representations of `collins` and `carter`, as well as `collins` and `clinton`
   - Choose the string with the better distance out of `carter` and `clinton`

— End of Test —