Family Name: . . . . . **Other Names:** . . . . . .

Signature: . . . . . **ID Number**: . . . . . .

COMP261: Test

10 Sept 2013

Instructions

- Time allowed: **50 minutes** .
- Answer **all** the questions. There are 50 marks in total.
- Write your answers in the boxes in this test paper and hand in all sheets.
- If you think some question is unclear, ask for clarification.
- This test contributes 15% of your final grade
- You may use paper translation dictionaries, and non-programmable calculators.
- You may write notes and working on this paper, but make sure your answers are clear.

Questions	**Marks**	
1. A* and Shortest Paths	[19]	
2. Z-buffer rendering	[19]	
3. Union Find	[12]	
	TOTAL:	

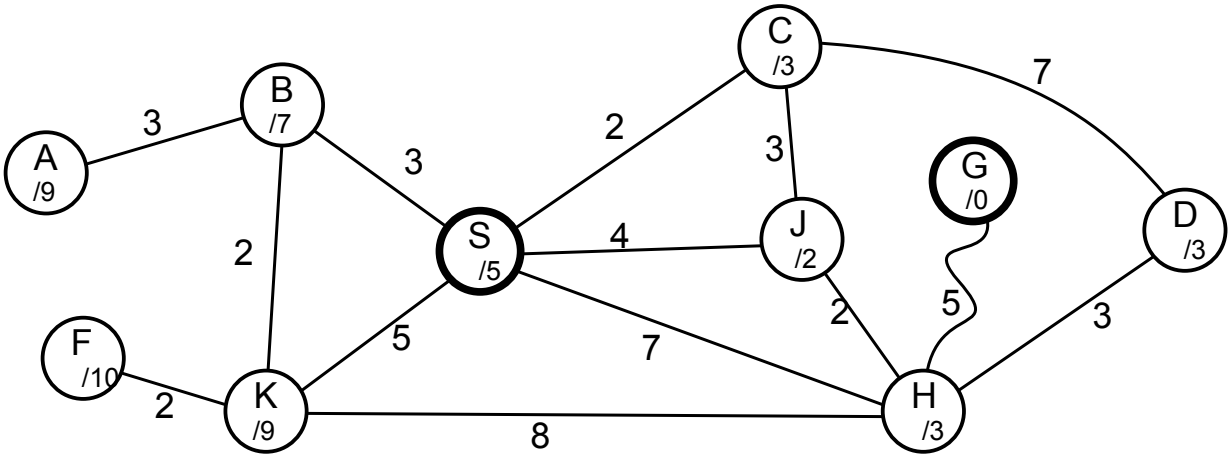
Question 1. Shortest Paths [19 marks]

(a) [9 marks] Show how A* finds the shortest path from **S** to **G** in the weighted undirected graph below, where the heuristic estimate of the cost from each node to the goal **G** is shown in the node. You may assume that the heuristic estimate is both admissable and consistent.

On the facing page, you should show the queue and the solution:

- list each element that is added to the priority queue, showing the node, from-node, cost to node, and total estimated cost.
  (The first element put on and removed from the priority queue is shown as an example.)
- add neighbours in alphabetical order
- indicate the order the elements are removed from the priority queue (eg, by numbering them)
- list the nodes of the shortest path found.

You should also show (on the graph below) all the information that is stored in the nodes of the graph during the search.



(Question 1 continued on next page)

(Question 1 continued)

Queue:

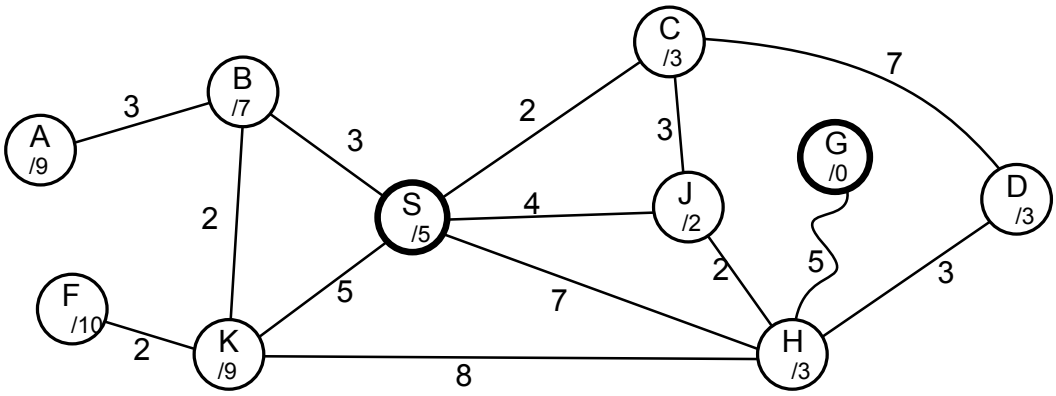
```
[Node ← From:  Cost, TotEst] When removed
[S    ← null:   0,   5     ]   #1
```

The Shortest Path Found:

(Question 1 continued on next page)

(Question 1 continued)

(Figure from previous page repeated)



The choice of the heuristic estimate is important for A*. Ideally, it should be

- admissable and
- consistent (the estimate never becomes less accurate along a path)

(b) [2 marks] Explain why the heuristic would not be admissible if it gave an estimated remaining cost of 9 for node J in the graph above.

(Question 1 continued on next page)

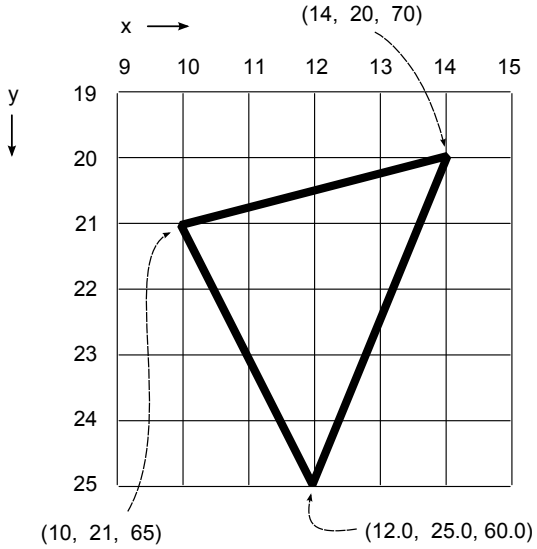
(Question 1 continued)

(c) [4 marks] Explain how an estimated remaining cost of 9 for node J would prevent A* from finding the shortest path.

(d) [4 marks] If the heuristic gave an estimated remaining cost of 7 for node C, it might still be admissible, but it would not be consistent. Explain why.

Question 2. Computer Graphics [19 marks]

(a) [6 marks] Show the values in the edgelists that would be constructed for the following polygon by the rendering algorithm described in the lectures. The (x,y,z) coordinates of the vertices are shown. State the order that you processed the edges and any assumptions you made.



Edge-Lists

y	x_{left}	z_{left}	x_{right}	z_{right}
20				
21				
22				
23				
24				
25				

Order edges processed:

Assumptions, if any

Working:

edge1: mx = mz =

edge2: mx = mz =

edge3: mx = mz =

(Question 2 continued on next page)

(Question 2 continued)

(b) [6 marks] Suppose your z-buffer rendering program has written two polygons (coloured red and green) into the z-buffer, and is now processing a new polygon, whose shading (reflected light colour) is blue. The background is black.

Some of the edgelists for the new (blue) polygon are:

y	left x	left z	right x	right z
20	10	29	15	24
21	11	32	15	28

The current contents of the Z-buffer are the following. Each cell of the Z-buffer contains a colour and a z value (z increases away from the viewer).

y \ x	10	11	12	13	14	15	16
20	black / ∞	black / ∞	green / 22	green / 24	green / 26	green / 30	black / ∞
21	red / 15	red / 15	black / ∞	green / 26	green / 30	green / 34	black / ∞

Show the contents of the Z-buffer after your program has rendered the edgelist data into the Z-buffer. State any assumptions you are making.

y \ x	10	11	12	13	14	15	16
20	/	/	/	/	/	/	/
21	/	/	/	/	/	/	/

Working:

for y = 20: mz =

for y = 21: mz =

(Question 2 continued on next page)

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

(Question 2 continued)

(c) [7 marks] When there are lots of objects in the scene, many of them hidden, the Z-buffer rendering algorithm may do lots of work rendering polygons into the Z-buffer that are then overwritten. This seems wasteful.

Consider two other approaches:

- Find all the polygons that are in front of the scene and render just those polygons.
- For each pixel in the image, find the polygon that is closest to the front at that pixel, and use it to render the pixel.

Explain why these approaches would not work, or would be less efficient than the Z-buffer rendering.

Question 3. Union Find

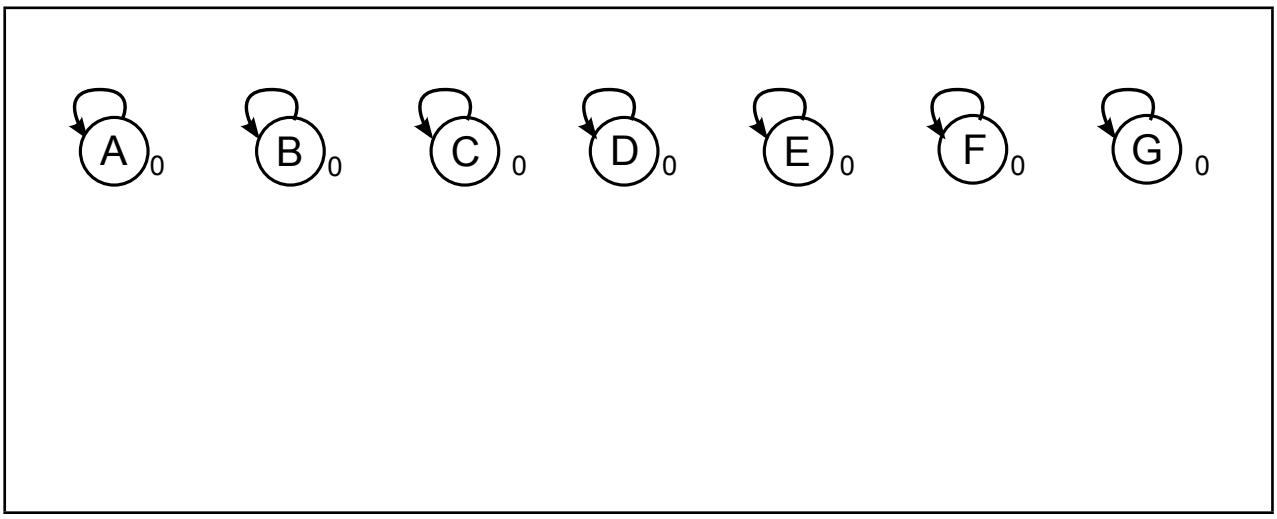
[12 marks]

(a) [4 marks] The following diagram shows a collection of sets of letters, represented using the union-find data structure. Initially, all the sets have just one element each. (The numbers are the ranks.)

Show the changes that will be made to the structure (including changes to the ranks of the nodes) if the following operations are applied, using the union-find algorithm below:

Union(A, B), Union(E, D), Union(C, B), Union(D, F)
Union(D, B), Union(F, C), Union(G, C)

Note that this version of the algorithm uses ranks (making the first argument of union be the parent if the ranks are equal) but does **NOT** change parents during the find.




```
Find(x):
    IF x.parent = x  THEN
        return x
    ELSE
        return Find(x.parent)
```

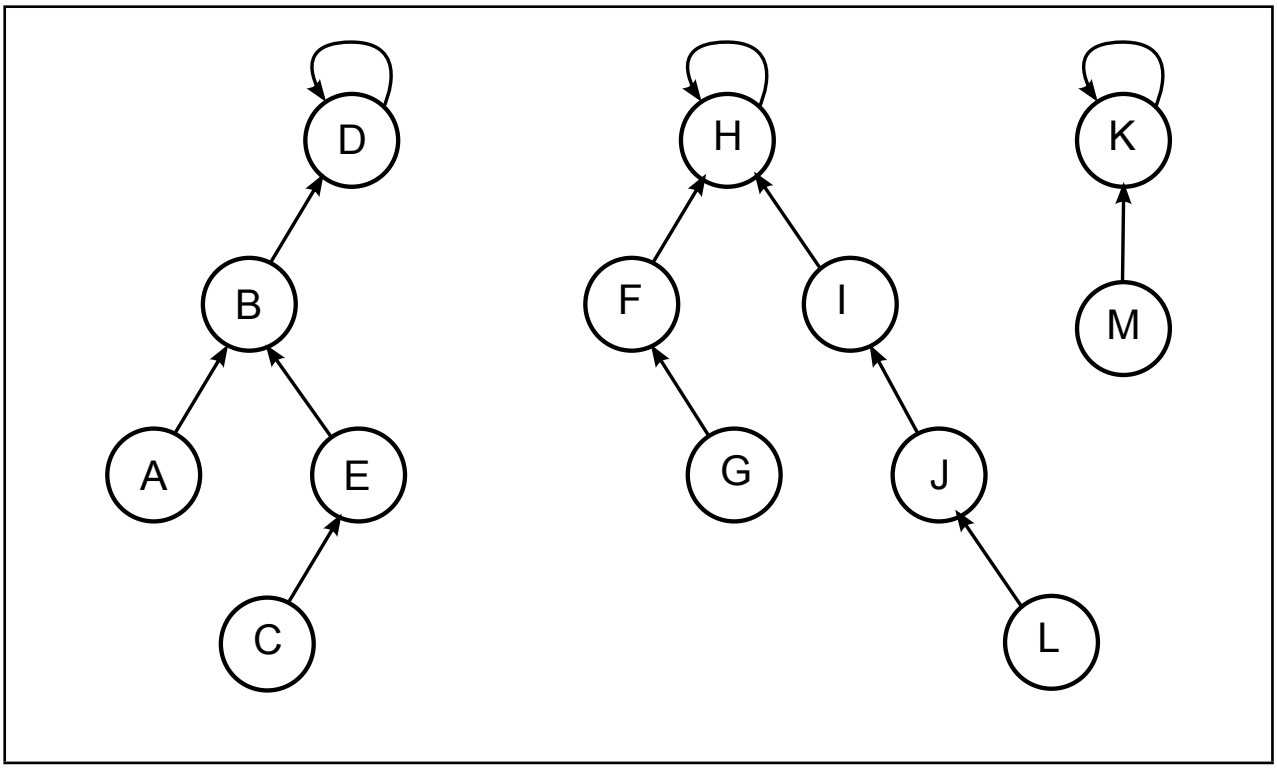
```
Union(x, y):
    xroot ← Find(x)
    yroot ← Find(y)
    IF xroot.rank < yroot.rank THEN
        xroot.parent ← yroot
    ELSE IF xroot ≠ yroot THEN
        yroot.parent ← xroot
        IF xroot.rank = yroot.rank THEN
            xroot.rank++
```

(Question 3 continued on next page)

(Question 3 continued)

(b) [4 marks] Show the changes that will be made to the Union-Find structure below if the following operations are applied, using the alternative union-find algorithm below which does **NOT** use ranks but does change parents during the find:

Union(G, A), Union(C, L)




```
Find(x):
    IF x.parent = x  THEN
        return x
    ELSE
        x.parent ← Find(x.parent)
        return x.parent
```

```
Union(x, y):
        xroot ← Find(x)
        yroot ← Find(y)
        yroot.parent ← xroot
```

(Question 3 continued on next page)

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

(Question 3 continued)

(c) [4 marks] Discuss the two versions of the Union-Find algorithm above. What is the advantage of each over the basic Union-Find algorithm. Which has the best performance and why?

****