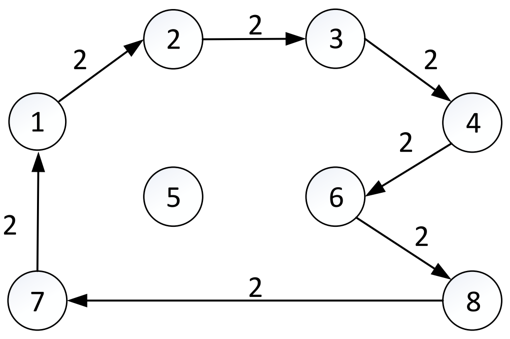
PART 1 – 15 MARKS - MULTIPLE CHOICE

Instructions
Please enter your answers on the bubble sheet with your name.
Use pencil only.
YOU CAN GUESS. MARKS WILL NOT BE DEDUCTED FOR FALSE ANSWERS.

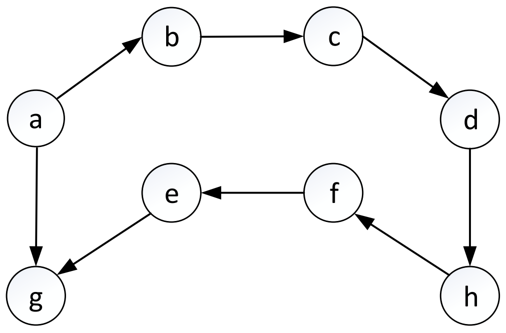
Questions

1. (2 marks) Is this graph:



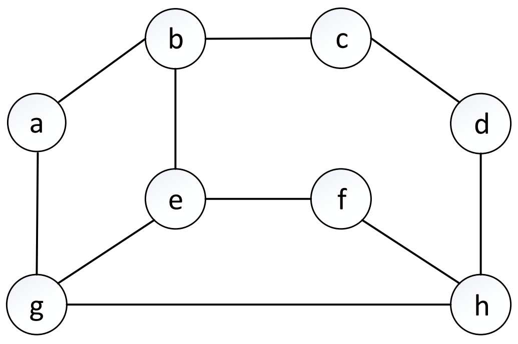
A. Weighted and connected
[✓ B. Weighted and disconnected]
C. Unweighted and connected
D. Unweighted and disconnected

2. (2 marks) Is this graph:



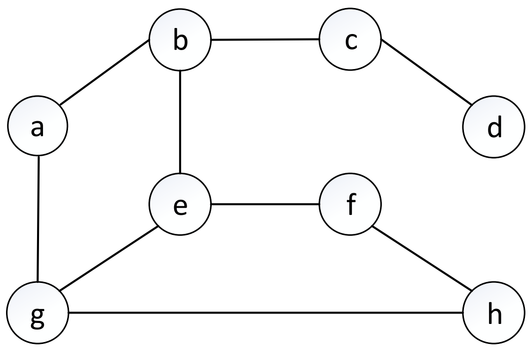
A. Directed and cyclical
[✓ B. Directed and acyclical]
C. Undirected and cyclical
D. Undirected and acyclical

3. (2 marks) Which one of these sequences of vertices **was not** generated from a BFS (Breadth First Search) traversal of the graph on the left? Assume ties are broken by the alphabetical order of the vertices.



A. a b g c e h d f
[✓ B. c b d a e f h g]
C. e b f g a c h d

4. (2 marks) Which one of these sequences of vertices **was** generated from a DFS (Depth First Search) traversal of the graph on the left? Assume ties are broken by the alphabetical order of the vertices.



A. a b c d g e f h
B. c b d a e g f h
[✓ C. e b a g h f c d]

5. (2 marks) Which is the worst (i.e. slowest) complexity for an algorithm:

   A. $\Omega(n)$

   B. $\Omega(n^2)$

   C. $\Omega(n \log_2 n)$

   [✓ D. $\Omega(n^2 \log_2 n)$]

6. (3 marks) What is the **exact cost** of this algorithm as a function of n:

```
myloops (n)
    for (i=n; i>=1; i--)
        for (j=i; j>=0, j--)
            // the basic operation is here
```

   A. $(n-1)(n+2)/2$

   B. $(n+1)^2/2$

   C. $(n+1)(n+2)/2$

   [✓ D. $n(n+3)/2$]

7. (2 marks) Do these two algorithms have the same asymptotic time cost::

```
exp1(a,n)
    result = 1
    for i=1 to n do
        result = result * a
    return result
```

```
exp2(a,n)
  if n=0     return 1
  if n=1     return a
  return     exp2(a, n div 2) *
             exp2(a, (n+1) div 2)
```

   [✓ A. Yes]

   B. No

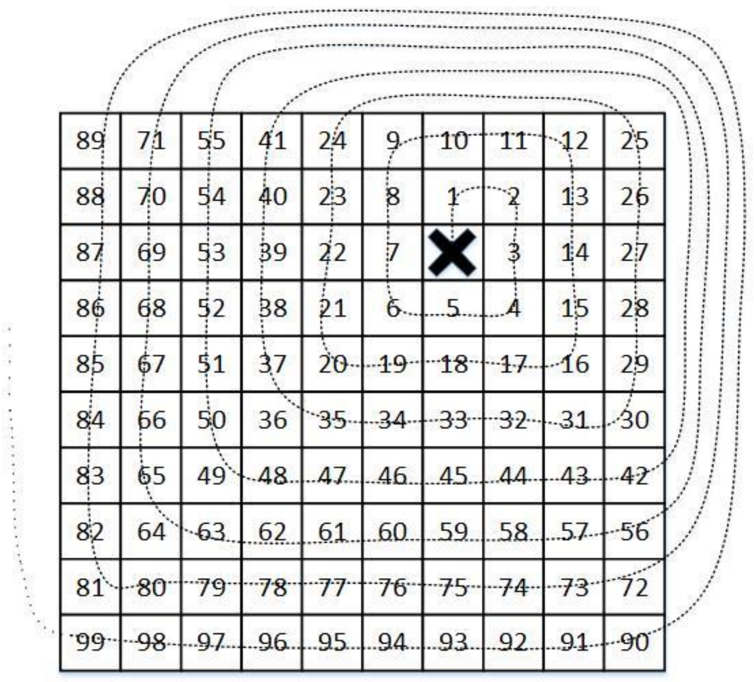
PART 2 – SHORT ANSWERS – 30 MARKS - PLEASE WRITE YOUR ANSWERS DIRECTLY IN THIS EXAM

8. Below is a mine sweeper algorithm to find the first closest mine to a cell in an n×n grid in a clockwise fashion around that cell starting at noon.

```
// This function returns the coordinates of the first occurrence of a mine
// in an nXn grid called   "grid"   around the cell at position (row,col)
// or NOMINE if there is no mine.
// The top left cell in the grid is at coordinates (1,1)
firstmine (row,col)
    found = NOMINE
    r = row
    c = col
    for level=1 to n-1 do
        r = r-1                   // go to row above last level
        while c ≤ col+level       visit(r,c++)  // move to the right (above cell)
        while r ≤ row+level       visit(r++,c)  // move downwards (right of cell)
        while c ≥ col-level       visit(r,c--)  // move to the left (below cell)
        while r ≥ row-level       visit(r--,c)  // move upwards (left of cell)
    return found

visit (r,c)
    if r<1 or r>n or c<1 or c>n   return
    if found ≠ NOMINE             return
    if grid(r,c) contains a mine  found = (r,c)
    return
```

For example, here is the order in which the cells around the ✖ (which is at coordinates (3,7)) would be visited in a 10×10 grid:



a) (1 mark) What is the **size** of this algorithm? (i.e. what variables will the cost be a function of?)

▶ n

b) (1 mark) Circle the **basic operation** of this algorithm which will be used to compute its complexity

In the remaining questions the **exact cost** refers to an exact function of the size, and the **asymptotic cost** refers to a O( ) function of the size. All costs are about **time**, not space.

c) (2 marks) What is the **exact best case** cost of this algorithm and when does it happen?

- ▶ All the loops always execute fully whether or not a mine is found, so the best, worst and average cases are all the same and they happen all the time.
- ▶ $4n(n-1) = 4n^2-4n$

d) (2 marks) What is the **exact worst case** cost of this algorithm and when does it happen?

- ▶ $4n^2-4n$ all the time

e) (1 marks) What is the **exact average case** cost of this algorithm?

- ▶ $4n^2-4n$ all the time

f) (5 marks) Make a **simple modification** (very few lines are affected) to this algorithm that will improve the best and average case costs at the same time, and will improve them in such a way that the exact best case cost will now be 1.

Note that you only need to provide pseudocode. This means that even though it should be possible to implement your pseudocode very much as you describe it, you do not have to make your code work for a specific programming language. Just make your intention very clear and unambiguous.

```
firstmine (row,col)
    found = NOMINE
    r = row
    c = col
    for level=1 to n-1 do
        r = r-1
        while c ≤ col+level     {visit(r,c++) ; if found ≠ NOMINE return found}
        while r ≤ row+level     {visit(r++,c) ; if found ≠ NOMINE return found}
        while c ≥ col-level     {visit(r,c--); if found ≠ NOMINE return found}
        while r ≥ row-level     {visit(r--,c) ; if found ≠ NOMINE return found}
    return found

visit (r,c)
    if r<1 or r>n or c<1 or c>n     return
    ~~if found ≠ NOMINE             return~~
    if grid(r,c) contains a mine    found = (r,c)
    return
```

- ▶ Note that it is not enough to break out of the while loops, you must return. Why?
- ▶ Note that a more elegant solution would be the redefine *visit* as a Boolean function *minefound* which indicates whether a mine has been found and have the test be *"if minefound(…"*. However this is not necessary from an efficiency point of view.
- ▶ Some languages may provide mechanisms for the function *visit* to break out of both itself and *firstmine* when it finds a mine. Depending on the implementation this may or may not be more efficient, but asymptotically it makes no difference.

g) (4 marks) **Explain** why your new algorithm is faster .

- ▶ Because it returns as soon as a mine is found. The searching is shortcircuited and the remaining mines are not visited.

h) (1 mark) When does the best case happen in the new algorithm?

- ▶ When there is a mine in the first cell which is the cell at location (row-1,col) i.e. directly above the cell around which we are circling.

i) (4 marks) Explain in English what additional changes you could make to the algorithm so that the worst case cost now becomes about a quarter of its original cost.

- ▶ The current algorithm is circling around the cell visiting $4n^2$-4n positions even though the size of the grid is only n^2. We should change the algorithm so that it only visits the cells in the grid, i.e. at most n^2-1 cells (not including the original cell).
- ▶ This is accomplished by moving the first if condition out of the function *visit* and integrating it in the 4 while loops to make sure that they **do not iterate** $4n^2$-4n times, i.e. each loop should stop as soon as it reaches a position that is outside of the grid. However, because the algorithm is traveling through the grid in a concentric pattern, it should position itself on the virtual corners of this pattern in order not to lose it.
- ▶ Finally, the simplest way to detect that there are no mines in the grid is to have a simple countdown of the visits.

j) (5 marks) Indicate in the code how you would reorganize it. You can write code if you want but you do not need to do so, you can just explain how you would adjust the components to get your improvements to the worst case cost.

- ▶ Note that the changes from question f) were integrated in the code.

```
firstmine (row,col)
    remainingvisits = n^2-1        // see k) for explanation of n^2-1
    found = NOMINE
    r = row
    c = col
    for level=1 to n-1 do
        r = r-1
        while r>0  and c ≤ col+level and c ≤ n
            {visit(r,c++) ; if found ≠ NOMINE return found}
        c = col + level
        while c ≤ n and r ≤ row+level and r ≤ n
            {visit(r++,c) ; if found ≠ NOMINE return found}
        r = row + level
        while r ≤ n and c ≥ col-level and c > 0
            {visit(r,c--); if found ≠ NOMINE return found}
        c = col - level
        while c > 0 and r ≥ row-level and r > 0
            {visit(r--,c) ; if found ≠ NOMINE return found}
        r = row – level
        if remainingvisits = 0 return found
    ~~return found~~

visit (r,c)
    remainingvisits--
    ~~if r<1 or r>n or c<1 or c>n            return~~
    if grid(r,c) contains a mine          found = (r,c)
    return
```

k) (3 marks) What would be the **exact worst case cost** of your newly modified algorithm and when would it happen?

- ▶ The worst case will now happen when there are no mines in the grid and all the cells are visited. Because the size of the grid is n^2 but the original cell is not visited, this will cost n^2-1

l) (1 mark) Would your modifications also improve the **asymptotic worst case cost** of your algorithm?

- ▶ No: the original exact worst case cost was $4n^2$-4n and the new exact worst case cost is n^2-1. Both are $O(n^2)$