University of Waterloo

University of Waterloo
Final Examination

Fall 2011

Student Name _____

Student Identification Number _____

Course Abbreviation & Number	CS 466/666
Course Title:	Advanced Algorithms
Section(s):	01
Section Combined Course(s):	none
Section Numbers of Combined Course(s):	none
Instructor:	T. Biedl
Date of Exam:	Monday, December 12, 2011
Time Period:	Start Time: 12:30pm End Time: 3:00pm
Duration of Exam:	2 hour 30 minutes
Number of Exam Pages	12 pages (includes cover page)
Exam Type	Special Material
Materials Allowed	1 letter-sized sheet of paper with anything written or typed on both sides.

Exams are printed double sided on white paper.

☐ Select this box if second side of paper is to be used for rough work calculations.

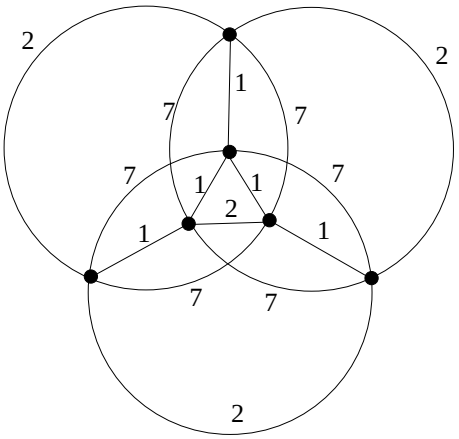
1. Complete all answers in the spaces provided.
2. Proctors will only confirm or deny the existence of errors on the exam.
3. All answers must be justified (unless specifically said so otherwise.)
4. If the question is not clear, state any assumption you make.
5. Cheating is an academic offence. Your signature on this exam indicates that you understand and agree to the university's policies regarding cheating on exams.

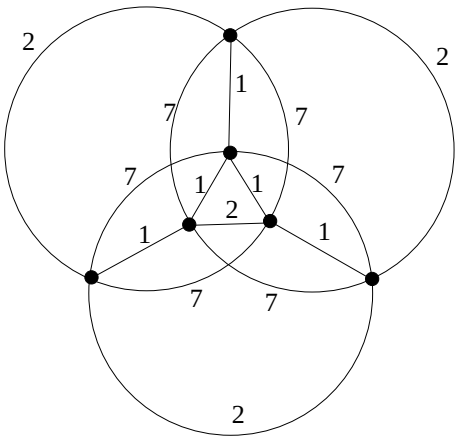
Signature: _____

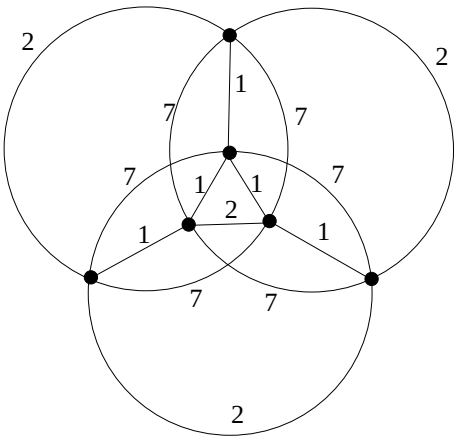
#	Marks	Actual	Initial
1	10		
2	18		
3	14		
4	10		
5	10		
6	10		
7	8		
8	12		
9	8		
$\sum$	100		

1. (10 marks): **Travelling Salesman Problem**

   (a) Apply the Christofides heuristic to the instance of Travelling Salesman given below. Show some intermediate steps. For your convenience, there are three copies of the graph. Clearly indicate which copy contains the final answer, i.e., the tour of the graph.







   (b) The optimum tour in this graph has length 8, while the tour you found in (a) should have had length more than 12. But in class we showed that the Christofides heuristic is a 1.5-approximation. Why is this not a contradiction?

2. (18 marks): **2-SAT and Max-2-SAT**
Consider the following set of 12 clauses:

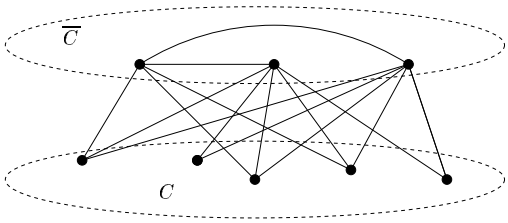
$c_1 : x_1 \vee x_2$ $c_2 : x_1 \vee x_3$ $c_3 : \overline{x_1} \vee x_4$ $c_4 : \overline{x_1} \vee \overline{x_4}$ $c_5 : x_2 \vee x_3$ $c_6 : x_2 \vee \overline{x_5}$

$c_7 : \overline{x_2} \vee x_3$ $c_8 : \overline{x_2} \vee \overline{x_5}$ $c_9 : x_3 \vee x_5$ $c_{10} : \overline{x_3} \vee x_4$ $c_{11} : \overline{x_3} \vee \overline{x_4}$ $c_{12} : \overline{x_4} \vee x_5$

(a) Argue that this instance of 2-SAT is not satisfiable.

(b) Give an assignment of boolean value to variables $x_1, \ldots, x_5$ such that the number of satisfied clauses is at least $\frac{3}{4}$ times the optimum number of satisfied clauses. Briefly explain how you obtained your assignment.

3. (14 marks): **Maximum Cut**
The *Maximum Cut* problem is the following problem: Given a graph $G = (V, E)$, find a set of vertices C such that the number of edges in the *cut*, i.e., the number of edges where one endpoint is in C and the other endpoint is not in C, is maximized. The figure below shows a graph with a cut of size 13.



(a) Consider the following randomized algorithm: For each vertex $v \in V$, add v to C with probability $\frac{1}{2}$. Prove that this is a $\frac{1}{2}$-approximation algorithm for maximum cut.

(b) Give a deterministic polynomial-time $\frac{1}{2}$-approximation algorithm for Maximum Cut.

4. (10 marks): **APX-hardness**
   Recall that *Clique* is the problem of finding a maximum set C of vertices in a graph $G = (V, E)$ such that any pair of vertices in C has an edge between them.

   Prove that Clique is APX-hard. You may use without proof that the following problems are APX-hard: Max3SAT, VertexCover, IndependentSet, TSP, Max2SAT($2L$).

5. (10 marks): **Dominating set**
   A *dominating set* in a graph $G = (V, E)$ is a set D of vertices such that for any vertex $v \in V$, either v or a neighbour of v is in D. The *dominating set problem* is the problem of finding a minimum dominating set.

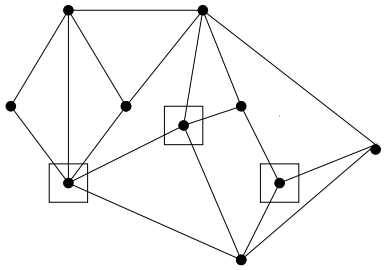
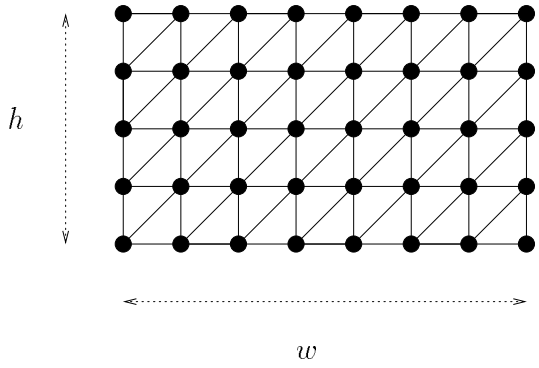


Figure 1: A graph with a dominating set D (boxed vertices.)

Give an algorithm that tests whether a graph G with maximum degree 5 has a dominating set of size k. Your algorithm should be fixed-parameter tractable in k. State the run-time of your algorithm. (You may use O^*-notation if you want.)

6. (10 marks): **Hexagonal grid graph**
Recall that a hexagonal grid graph of height h and width w is any graph G that is a subgraph of the following graph:



(a) Argue that computing the size of a minimum vertex cover is polynomial in hexagonal grid graphs of height h if h is a constant.
(Hint: This is meant to be easy.)

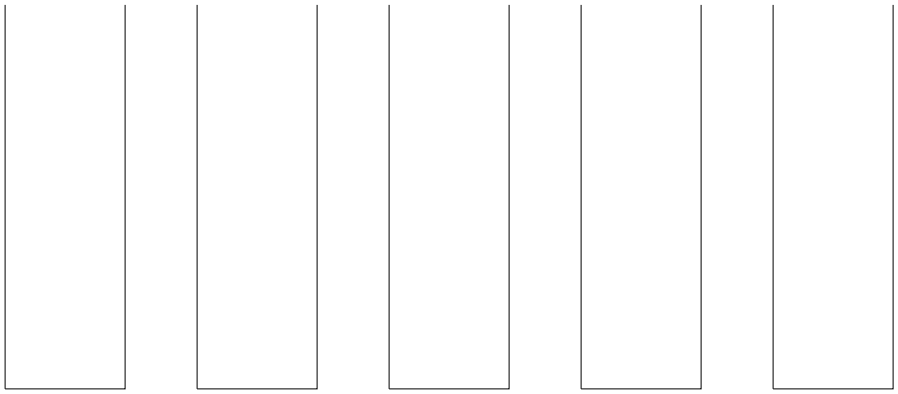
(b) Give a PTAS for minimum vertex cover in hexagonal grid graphs. It suffices to sketch the main ideas; you need not give all details.

7. (8 marks): **Bin packing**

   (a) Apply the first-fit algorithm to pack the following items into bins of size 1:

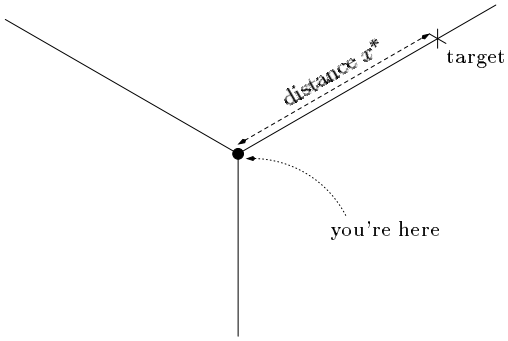
   $$\frac{38}{240}, \frac{81}{240}, \frac{39}{240}, \frac{37}{240}, \frac{82}{240}, \frac{121}{240}, \frac{80}{240},$$

   Your drawing need not be to scale, but indicate clearly which item has been placed in which bin. You may not need all bins.



   (b) Does the packing you obtain use the minimum possible number of bins? Justify your answer.

8. (12 marks): **Lost cow problem**



Consider a variant of the lost cow problem where there are three possible ways to go. Thus, you are standing at a point from which there are three paths P_1, P_2, P_3. On one at the paths (you don't know which one), at distance x^* (you don't know x^*, but $x^* \geq 1$), there is a target that you need to find.

Give a strategy to find the target. Argue that your strategy is c-competetive for some constant c. (Any constant will give you full credit; some bonus marks may be had if you make c especially small.)

9. (8 marks): **Online matching problem**
   The *online matching problem* is defined as follows. You are given a graph in an online fashion as follows: Initially the graph is empty. Each next step either adds a vertex, or it adds an edge for which both endpoints had been added previously. Every time an edge e is added, you need to decide whether to include e in a set M (which was initially empty.) M must be a matching at all times. Once an edge has been added to M, it cannot be removed from it.

   Give an algorithm for the online matching problem that finds a matching of at least half the size of a maximum matching in the final graph. Justify your answer.