Sample midterm

1. (3 points) Write a first order logic formula which is equivalent to the following but all ¬ symbols appear immediately in front of a predicate.

$$\neg\forall a((\exists b\ p(a,b) \rightarrow \exists c\ p(a,c)) \wedge \neg\forall d\ q(a,d))$$

   Here p and q are predicates.

2. (5 points) Prove the following statement without using Theorem 1

   Let G be a multigraph. If G has an Eulerian circuit and G has no zero degree vertex then G is a connected and all vertices of G have even degree.

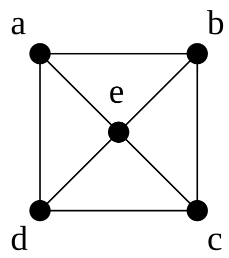
3. (12 points) Prove the following statement without using Dirac's theorem (Theorem 3).

   If a graph G has at least 3 vertices and the degree of every vertex of G is at least $\frac{|V(G)|}{2}$ then G has a Hamiltonian cycle.

4. (7 points) Prove that every hypercube Q_n with $n \geq 2$ has a Hamiltonian cycle.

5. (3 points) (Rosen, p.646, q47c))

   Determine if this graph contains a Hamiltonian cycle. If does, write down the **vertices** visited by your circuit in the order they are visited (no justification needed in this case). If it does not, give a reason why.



6. (10 points)

   Prove that Kruskal's algorithm produces a minimum (weight) spanning tree without using Theorem 5. You may assume that the output of Kruskal's algorithm is a tree (and any optimal output is a tree).

Appendix 1: Rules of inference

Rule		Name
$\dfrac{P \wedge Q}{P}$	$\dfrac{P \wedge Q}{Q}$	$\wedge\mathcal{E}$
$\dfrac{\begin{array}{c}P\\ Q\end{array}}{P \wedge Q}$	$\dfrac{\begin{array}{c}P\\ Q\end{array}}{Q \wedge P}$	$\wedge\mathcal{I}$
$\dfrac{\begin{array}{c}\cancel{P}\\ \vdots\\ Q\end{array}}{P \to Q}$		$\to\mathcal{I}$
$\dfrac{\begin{array}{c}P\\ P \to Q\end{array}}{Q}$		$\to\mathcal{E}$
$\dfrac{P}{P \vee Q}$	$\dfrac{P}{Q \vee P}$	$\vee\mathcal{I}$
$\dfrac{\begin{array}{c}P \vee Q\\ P \to R\\ Q \to R\end{array}}{R}$	$\dfrac{\begin{array}{c}Q \vee R\\ P \to R\\ Q \to R\end{array}}{R}$	$\vee\mathcal{E}$
$\dfrac{P \to \mathbf{F}}{\neg P}$		$\neg\mathcal{I}$
$\dfrac{\begin{array}{c}P\\ \neg P\end{array}}{\mathbf{F}}$		$\neg\mathcal{E}$
$\dfrac{\neg\neg P}{P}$		$\neg\neg\mathcal{E}$
$\dfrac{\mathbf{F}}{P}$		$\mathbf{F}\mathcal{E}$

Appendix 2: Table of equivalences

For propositional logic.

$p \wedge \mathbf{T} \equiv p$ $p \vee \mathbf{F} \equiv p$
$p \vee \mathbf{T} \equiv \mathbf{T}$ $p \wedge \mathbf{F} \equiv \mathbf{F}$
$p \vee p \equiv p$ $p \wedge p \equiv p$
$\neg(\neg p) \equiv p$
$p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$
$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$ $(p \vee q) \vee r \equiv p \vee (q \vee r)$
$\neg(p \vee q) \equiv \neg p \wedge \neg q$ $\neg(p \wedge q) \equiv \neg p \vee \neg q$
$p \vee (p \wedge q) \equiv p$ $p \wedge (p \vee q) \equiv p$
$p \vee \neg p \equiv \mathbf{T}$ $p \wedge \neg p \equiv \mathbf{F}$
$p \rightarrow q \equiv \neg p \vee q$

For first order logic. The above as well as the following.

$\neg \exists x \ p(x) \equiv \forall x \ \neg p(x)$ $\neg \forall x \ p(x) \equiv \exists x \ \neg p(x)$

Appendix 3: Definitions and theorems

Definition 1. We call $P_1, \ldots, P_k \vdash Q$ an **argument**. An argument is **valid** if we can infer the conclusion Q given the hypotheses $P_1, \ldots, P_k$ and **invalid** otherwise.

Definition 2. A **graph** G is an ordered pair (V, E) where V is a set of vertices and E is a (multi) set of edges: 2-element subsets of V.

Definition 3. A **walk** consists of an alternating sequence of vertices and edges consecutive elements of which are incident, that begins and ends with a vertex. A **trail** is a walk without repeated edges. A **path** is a walk without repeated vertices.

If a walk (resp. trail, path) begins at x and ends at y then it is an $x - y$ walk (resp. $x - y$ trail, resp. $x - y$ path).

A walk (trail) is **closed** if it begins and ends at the same vertex. A closed trail whose origin and internal vertices are distinct is a **cycle**.

Definition 4. A **circuit** is a trail that begins and ends at the same vertex.

Some equivalent definitions of paths and cycles.

Definition 5. A **path** in a graph G is a sequence of vertices $p_1, \ldots, p_k$ such that for all $1 \leq i \leq k-1$, (p_i, p_{i+1}) is an edge in G.

A **cycle** in a graph G is a path $p_1, \ldots, p_k$ such that (p_k, p_1) is an edge of G.

Definition 6. A **subgraph** H of a graph G is a graph such that $V(G) \subseteq V(G)$ and $E(H) \subseteq \{(u,v) | (u,v) \in E(G), u \in V(H), v \in V(H)\}$.

An **induced subgraph** H of G is a subgraph of G where $E(H) = \{(u,v) | (u,v) \in E(G), u \in V(H), v \in V(H)\}$ (i.e., we have all edges between vertices of H).

Definition 7. P_n is the graph on n vertices $v_1, \ldots, v_n$ and edges (v_i, v_{i+1}) for each i from 1 to $n-1$.

C_n is the graph on n vertices $v_1, \ldots, v_n$ and edges (v_1, v_n) and (v_i, v_{i+1}) for each i from 1 to $n-1$.

K_n, the complete graph, is the graph on n vertices $v_1, \ldots, v_n$ and all edges (i.e., (v_i, v_j) for all $1 \leq i < j \leq n$).

Q_n, the hypercube graph, is the graph on 2^n vertices with each vertex labelled by a different binary string of length n and two vertices are adjacent if and only if their labels in exactly one bit.

Definition 8. A **path** in a graph G is a subgraph of G that is a copy of P_k for some k

A **cycle** in a graph G is a subgraph of G that is a copy of C_k for some k

Definition 9. The **length** of a path P is the number of vertices in it and is denote $|P|$ or $|V(P)|$. The **length** of a cycle is the number of vertices in it.

Definition 10. An **Eulerian circuit** in a graph G is a circuit which contains every edge of G.

An **Eulerian trail** in a graph G is a trail which contains every edge of G.

Definition 11. A graph G is **connected** if there is a path between every pair of vertices. G is **disconnected** otherwise.

A graph G is **k-connected** if there does not exist a set of at most $k-1$ vertices of G whose removal yield a disconnected graph.

A **connected component** of a graph G is a maximal connected subgraph (meaning we cannot add more edges and vertices while preserving connectivity).

Theorem 1. *Let G be a multigraph. G is a connected and all vertices of G have even degree if and only if G has an Eulerian circuit and G has no zero degree vertex.*

Definition 12. An **Hamiltonian cycle** in a graph G is a cycle which contains every vertex of G.

An **Hamiltonian path** in a graph G is a path which contains every vertex of G.

Theorem 2. *There exists an ordering (or sequence) containing all n-bit binary strings exactly once where every consecutive string differ in exactly one bit and the first and last string differ in exactly one bit.*

Namely, **Gray codes** *provide such an ordering.*

Lemma 1. *If a graph G has a Hamiltonian cycle then G is 2-connected.*

Theorem 3 (Dirac's theorem)**.** *If a graph G has at least 3 vertices and the degree of every vertex of G is at least $\frac{|V(G)|}{2}$ then G has a Hamiltonian cycle.*

Definition 13. A **tree** is a connected graph with no cycles.

A **forest** is a graph with no cycles (which is not necessarily connected).

Definition 14. A **rooted tree** is digraph obtained from a tree T and a special vertex $r \in V(T)$ called the **root** by directing every edge "towards" the root (e.g., from the vertex farthest from the root to the vertex closest to the root).

Lemma 2. *If T is a tree with at least 2 vertices then T has a vertex of degree 1.*

Theorem 4. *Every tree on n vertices has exactly $n-1$ edges.*

Problem 1. Minimum spanning tree

Input: A connected graph $G = (V, E)$ and weights $w_e \geq 0$ for each edge $e \in E$ **Output:** A subset F of E such that (V, F) is connected and given these restrictions, $\sum_{e \in E} w_e$ is maximized.

Algorithm 1. Kruskal's algorithm

Initialize F to the empty set.
Sort the edges in ascending order of weights
For each edge e in this ordering.
  If $(V, F \cup \{e\})$ does not contain a cycle then add e to F
Return F

Theorem 5. *Kruskal's algorithm returns a minimum spanning tree.*

Problem 2. Shortest path

Input: A connected graph $G = (V, E)$, weights $w_e > 0$ for each edge $e \in E$ and two vertices $s, t \in V$. **Output:** A minimum weight path from s to t in G.

Algorithm 2. (Simplified) Dijkstra's algorithm

Initialize an array d indexed by V to ∞
$d[s] \leftarrow 0$
$S \leftarrow \{s\}$
Initialize an array prev indexed by V to null.
While $t \notin S$
  Find $e = (u, v) \in E$ with $u \in S, v \in V \setminus S$ minimizing $d[u] + w_{(u,v)}$.
  $d[v] \leftarrow d[u] + w_{(u,v)}$
  $\text{prev}[v] \leftarrow u$
  $S \leftarrow S \cup \{v\}$
Return d and prev

To obtain the path from the output, repeatedly follow the prev pointers, starting from t.

Lemma 3. *Dijkstra's algorithm assigns d values in a non-decreasing order.*

Lemma 4. *A subpath of a minimum weight path is a minimum weight path (between different endpoints).*

Theorem 6. *The d values returned by Dijkstra's algorithm corresponds to minimum weight distance from s.*

Some more lemmas and theorems from the assignments.

Lemma 5. *Let G be a graph. If $C = c_1, c_2, \ldots, c_{k-1}, c_k$ is a cycle in G then for any j (between 1 and k), $c_j, c_{j+1}, \ldots, c_{k-1}, c_k, c_1, c_2, \ldots, c_{j-2}, c_{j-1}$ is also a cycle in G.*

Theorem 7. (Ore's theorem)
Let G be a graph. If G has at least 3 vertices and for every pair of non-adjacent vertices $u, v \in V(G)$, $\deg(u) + \deg(v) \geq |V(G)|$ then G has a Hamiltonian cycle.

Lemma 6. *Let G be a graph. For any $k > 2$, if G is k-connected then G is $k-1$ connected.*

Definition 15. A set of path $P_1, \ldots, P_k$ with the same starting and ending vertex is said to be **internally vertex disjoint** if no two paths have a vertex in common except for their endpoints. That is, if $P_i = u, p_{i,1}, p_{i,2}, \ldots, v$ then there does not exist i, j, k, ℓ with $i \neq k$ such that $p_{i,j} = p_{k,\ell}$.

Theorem 8. (Part of Menger's theorem)
Let G be a graph. If every pair of (distinct) vertices $u, v \in V(G)$, there are two vertex disjoint paths P_1, P_2 starting at u and ending at v then G is 2-connected.

Definition 16. The **Cartesian product** of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, denoted $G_1 \times G_2$, is a graph with vertex set V and edge set E defined as follows. V consists of all pair (v_1, v_2) for each vertex v_1 in V_1, and each vertex v_2 in V_2 (i.e., $V = \{(v_1, v_2) | v_1 \in V_1, v_2 \in V_2\}$). Two vertices (u_1, u_2) and (v_1, v_2) of $G_1 \times G_2$ are adjacent if either

- $u_1 = v_1$ and u_2 is adjacent to v_2 in G_2, or
- $u_2 = v_2$ and u_1 is adjacent to v_1 in G_1.

In other words, $E = \{((u_1, u_2), (v_1, v_2)) | u_1 = v_1, (u_2, v_2) \in E(G_2)\} \cup \{((u_1, v_1), (u_2, v_2)) | u_2 = v_2, (u_1, v_1) \in E(G_1)\}$.

Appendix 4: Glossary of symbols

Symbol	Name	Example or definition	Example read as
$\vee$	Logical or	$p \vee q$	p or q.
$\wedge$	Logical and	$p \wedge q$	p and q.
$\neg$	Logical not	$\neg p$	not p.
$\rightarrow$	Implication	$p \rightarrow q$	p implies q.
			If p then q.
			q whenever p.
$\leftrightarrow$	Bi-implication	$p \leftrightarrow q$	p if and only if q.
$\equiv$	Equivalence	$p \equiv q$	p is equivalent to q.
F	Contradiction	$\mathbf{F} \rightarrow p$	False implies p.
T	Tautology	$\mathbf{T} \rightarrow \mathbf{F}$	True implies false.
$\vdash$	Infer	$P_1, \ldots, P_k \vdash Q$	We can infer Q from $P_1, \ldots, P_k$.
$\models$	Models	$P_1, \ldots, P_k \models Q$	$P_1, \ldots, P_k$ models Q.
$\in$	Containment	$x \in S$	x is in S.
			x is an element of S.
$\cap$	Intersection	$S \cap T = \{x \vert x \in S, x \in T\}$	S intersect T.
			The elements in both S and T.
$\cup$	Union	$S \cap T = \{x \vert x \in S \text{ or } x \in T\}$	S union T.
			The elements in either S or T.
$\setminus$	Set difference	$S \setminus T = \{x \vert x \in S, x \notin T\}$	S minus T.
			The elements in S but not T.
$\forall$	Universal quantifier	$\forall x \in \mathbb{Z}, x^2 \geq 0$	For all integers x, x^2 is greater or equal to zero.
$\exists$	Existential quantifier	$\exists x \in \mathbb{Z}, x + 5 = 0$	There exists an integer x such that $x + 5$ is zero.