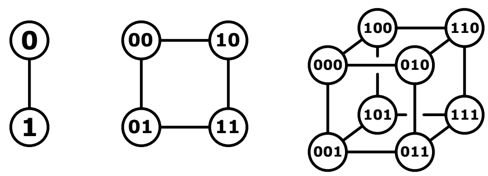
You have 180 minutes to answer six of these questions.
Write your answers in the separate answer booklet.

1. The d-dimensional hypercube is the graph defined as follows. There are 2^d vertices, each labeled with a different string of d bits. Two vertices are joined by an edge if and only if their labels differ in exactly one bit.



The 1-dimensional, 2-dimensional, and 3-dimensional hypercubes.

   (a) *[8 pts]* Recall that a Hamiltonian cycle is a closed walk that visits each vertex in a graph exactly once. ***Prove*** that for all $d \geq 2$, the d-dimensional hypercube has a Hamiltonian cycle.

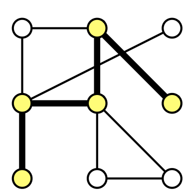
   (b) *[2 pts]* Recall that an Eulerian circuit is a closed walk that traverses each edge in a graph exactly once. Which hypercubes have an Eulerian circuit? *[Hint: This is very easy.]*

2. The University of Southern North Dakota at Hoople has hired you to write an algorithm to schedule their final exams. Each semester, USNDH offers n different classes. There are r different rooms on campus and t different time slots in which exams can be offered. You are given two arrays $E[1..n]$ and $S[1..r]$, where $E[i]$ is the number of students enrolled in the ith class, and $S[j]$ is the number of seats in the jth room. At most one final exam can be held in each room during each time slot. Class i can hold its final exam in room j only if $E[i] < S[j]$.

   Describe and analyze an efficient algorithm to assign a room and a time slot to each class (or report correctly that no such assignment is possible).

3. What is the *exact* expected number of leaves in an n-node treap? (The answer is obviously at most n, so no partial credit for writing "$O(n)$".) *[Hint: What is the probably that the node with the kth largest key is a leaf?]*

4. A *tonian path* in a graph G is a simple path in G that visits more than half of the vertices of G. (Intuitively, a tonian path is "most of a Hamiltonian path".) ***Prove*** that it is NP-hard to determine whether or not a given graph contains a tonian path.



A tonian path.

5. A *palindrome* is a string that reads the same forwards and backwards, like x, pop, noon, redivider, or amanaplanacatahamayakayamahatacanalpanama. Any string can be broken into sequence of palindromes. For example, the string bubbaseesabanana ('Bubba sees a banana.') can be broken into palindromes in several different ways; for example,

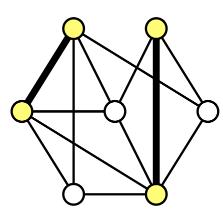
   bub + baseesab + anana
   b + u + bb + a + sees + aba + nan + a
   b + u + bb + a + sees + a + b + anana
   b + u + b + b + a + s + e + e + s + a + b + a + n + a + n + a

   Describe and analyze an efficient algorithm to find the smallest number of palindromes that make up a given input string. For example, given the input string bubbaseesabanana, your algorithm would return the integer 3.

6. Consider the following modification of the 2-approximation algorithm for minimum vertex cover that we saw in class. The only real change is that we compute a set of edges instead of a set of vertices.

```
ApproxMinMaxMatching(G):
  M ← ∅
  while G has at least one edge
      (u, v) ← any edge in G
      G ← G \ {u, v}
      M ← M ∪ {(u, v)}
  return M
```

   (a) *[2 pts]* ***Prove*** that the output graph M is a *matching*—no pair of edges in M share a common vertex.
   (b) *[2 pts]* ***Prove*** that M is a *maximal* matching—M is not a proper subgraph of another matching in G.
   (c) *[6 pts]* ***Prove*** that M contains at most twice as many edges as the *smallest* maximal matching in G.



The smallest maximal matching in a graph.

7. Recall that in the standard maximum-flow problem, the flow through an edge is limited by the capacity of that edge, but there is no limit on how much flow can pass through a vertex. Suppose each vertex v in our input graph has a capacity $c(v)$ that limits the total flow through v, in addition to the usual edge capacities. Describe and analyze an efficient algorithm to compute the maximum (s, t)-flow with these additional constraints. *[Hint: Reduce to the standard max-flow problem.]*