1. (a) (28 marks) In class we studied the use of consistent hashing in Chord for finding keys in peer-to-peer networks. Assume that each processor $p_i$ has a finger table that allows it to store its own address plus the addresses of two other processors. Which addresses must be stored in the table so as to, both, minimize the number of messages needed to find a key (or to decide that a key is not stored in the system), and to ensure that every key can be found?

You must explain your answer and show that the number of messages is minimized.

(b) (5 marks) Write in pseudocode an algorithm that uses your choice of fingers to find a document with a given key.

(c) (6 marks) Show that with your choice of fingers any key stored in the system can be found.

(d) (6 marks) Compute the number of messages that need to be sent to find a key in the worst case.

2. (20 marks) The Chord scheme that we have discussed in class to find information in a peer-to-peer network makes use of two hash functions: $h_p(id)$ that maps processor addresses to ring identifiers and $h_k(k_i)$ which maps document keys to ring identifiers. Assume that a new processor $p_i$ joins a peer-to-peer network, and the ring identifier assigned to $p_i$ is the same as the ring identifier of another processor $p_j$ in the network, i.e. $h_p(p_i) = h_p(p_j)$. To maintain load balancing assume that half of the documents stored in $p_j$ are moved to the new processor $p_i$.

Modify the find and help_find algorithms discussed in class, so that they still work in this case, i.e. if some processor $p_r$ requests a copy of a document stored in $p_i$ or $p_j$, the algorithms will correctly return it to $p_r$.

3. (5 marks) Compute the (simple) page rank of every node in the following graph. Indicate which method you used to compute the page rank.

4. (5 marks) Compute the stationary distribution of the following Markov chain. Explain how you computed the stationary distribution.

5. In class we talked about the problem of ranking pages to be returned by web search engines. Ideally high-quality content pages should be assigned high ranks, and low-quality content pages should be assigned low ranks.

Let $r(i)$ denote the page rank of page $i$, and let $n$ be the total number of pages in the Web graph. Consider the following 3 ways of defining $r(i)$:
a. (5 marks) $r(i) = 1/n$.

b. (10 marks) $r(i) = \max \{ r(j) \mid j \in B(i) \}$, where $B(i)$ is the set of pages that have hyperlinks pointing to page $i$.

c. (10 marks) $r(i) = \frac{1}{n} + \sum_{j \in B(i)} r(j)$.

d. (Optional 5 marks) $r(i) = \max \{ r(j) \mid j \in B(i) \} + \min \{ r(j) \mid j \in B(i) \}$, where $B(i)$ is the set of pages that have hyperlinks pointing to page $i$. If $B(i)$ has only one page $j$, then $\max \{ r(j) \mid j \in B(i) \} = \min \{ r(j) \mid j \in B(i) \} = r(j)$.

For this question you will assume that the Web Graph is strongly connected. For each one of the above methods indicate if (i) the page rank always exists and (ii) the method achieves the goal of assigning high rank to high-quality pages and low rank to low-quality pages. Explain your answers. If a rank function does not work, construct an example showing that either the page rank cannot be computed or that the rank function does not correctly differentiate between high quality content and low quality content pages. Assume that a low quality page has very few references to it and a high quality page has many references to it.

Recall that the sum of page ranks of all the pages must be 1.