Name: ________________________________

SID: ________________________________

- You may not use any reference materials during this exam.
- Electronic devices, including calculators, cell phones, mp3 players, and laptops are all prohibited.
- You may not use your own scratch paper. The exam has plenty and you can ask for more if needed.
- You may not leave the classroom once the exam has been distributed.
- Communicating with other students in any way is prohibited.

Academic Honesty: I understand that if I cheat on this exam in any way, I will receive the maximum possible penalty, including an F in this course.

Signature: ________________________________

<table>
<thead>
<tr>
<th>Problem</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/ 10</td>
</tr>
<tr>
<td>2</td>
<td>/ 15</td>
</tr>
<tr>
<td>3</td>
<td>/ 10</td>
</tr>
<tr>
<td>4</td>
<td>/ 20</td>
</tr>
<tr>
<td>5</td>
<td>/ 15</td>
</tr>
<tr>
<td>Total</td>
<td>/ 70</td>
</tr>
</tbody>
</table>
1 \( B^\epsilon \)-trees and balanced workloads

(10 points) Suppose we have a workload that is evenly split between inserts and lookups. Show that there exists a choice of \( \epsilon \) so that the time required to perform \( N \) inserts and \( N \) lookups in a \( B^\epsilon \)-tree will take less time than in a \( B \)-tree. To simplify things, you may assume the constants in the Big-Oh notation are all 1, e.g. that inserting into a B-tree requires exactly \( \log_B N \) I/Os.
2 De-amortized $B^\epsilon$-trees

Recall the $B^\epsilon$-tree variant from HW3 in which each internal node has a single buffer that can hold $B$ elements. When a node, $n$, becomes full, we partition its buffer into sub-buffers $S_1, S_2, \ldots, S_k$, where the elements in $S_i$ are destined for the $i$th child of $n$. We then flush the largest $S_i$ to its corresponding child.

Consider a slight variant where we flush at most $B^{1-\epsilon}$ elements of $S_i$ to the child.

(5 points) Prove that, if the child does not have enough space to receive all $B^{1-\epsilon}$ elements being flushed from its parent, that we can free up enough space by flushing a single sub-buffer in the child.

(5 points) What is the amortized running time for inserts in this $B^\epsilon$-tree?
(5 points) What is the worst-case time for an insert in this $B^\epsilon$-tree?
3 Avoiding left turns

(10 points) Suppose we want to find the shortest path from a node $s$ to a node $t$ in an undirected graph $G$, but with the following extra constraint: we want to avoid left turns. In other words, imagine the edges attached to each node are ordered, and a path that arrives on the $i$th edge and departs along the $j$th edge incurs a cost of $j - i + \text{deg}(v) \mod \text{deg}(v)$, as illustrated below.

Show how to compute shortest paths in this setting.
4 Skip lists with size information

Consider adding a field, size, to each node in a skip list. The meaning of size is as follows: If two consecutive nodes \( n_1 \) and \( n_2 \) in the same level have \( \text{size}_1 \) and \( \text{size}_2 \), respectively, then the number of elements in the bottom level strictly between \( n_1 \) and \( n_2 \) is \( \text{size}_1 \).

(10 points) Show how to maintain the size fields when inserting a new element in the skip list. (You don’t need to provide pseudo-code or a proof – a picture with a couple of sentences making your idea clear is sufficient).

(10 points) Explain how to find the \( i \)th largest element of the skip list using the size fields. Pseudo-code is probably the simplest way to explain this algorithm clearly.
5 In-memory B-trees

Consider using a B-tree as an in-memory data structure. In this case, we can choose any block size we want. This question tries to answer: what block size should we use?

- (5 points) What is the computational complexity of a lookup?
- (5 points) What is the worst-case computational complexity of an insert?
- (5 points) What block size should we use?