CS 240: Data Structures and Data Management **Spring 2023**

Tutorial 09: July 17

1. **Cycle in Cuckoo Hashing**
   Assume $M = 5$, $U = \{0, 1, 2, \ldots, 23, 24\}$ and suppose we are given following hash functions.

   $$h_0(k) = k \mod 5$$
   $$h_1(k) = \lfloor \tfrac{k}{5} \rfloor \mod 5$$

   Find a scenario where we end up with infinite loop after calling an `insert` (assume that our "emergency break" of $2n$ iterations does not exist) in Cuckoo Hashing. To be more specific, come up with the keys that are in each table and find a newly inserted key that will cause infinite loop.

2. **More on the Cycle in Cuckoo Hashing**
   Let's refer to back to algorithm for Cuckoo Hashing insertion and notice we only try $2n$ iterations. Why is it that we can stop after $2n$ iteration and return "failure to insert"?

3. **Programming Question: Trie Searching**
   Given a Standard Trie with branches of 0, 1, $, program a searchK function where the function will recursively report all strings in the Trie of length K, in no particular order, delimited by a newline between each string. For instance, if K = 3, "000$" will be one such string. All strings ends with $. The functions and classes you can use is given in the snippet below. You may use any helper function inside or outside of the Trie Class as needed.

```
#include <string>
#include <iostream>
// ... include libraries as needed

using namespace std;

class Trie{ // Uncompressed Trie
    private:
    // ... Implementation is omitted
    public:
    Trie* getChildA(); // returns the pointer to child A of the current trie
                       // return nullptr if doesn't exist
    Trie* getChildB(); // returns the pointer to child A of the current trie
                       //return nullptr if doesn't exist
    Trie* getChildDollar(); // returns the pointer to child $ of the current trie
                            //return nullptr if doesn't exist
    int getHeight(); // returns the current height (root is 0)

    void searchK(int K, string s); // Implement this as a class function
    // Hint: You can use s to store string built from traversal
};

void Trie::searchK(int K, string s){
```

```
    // put your code here
}

// ...
int main(){
    //......
    Trie root = Trie(); // Construct a root trie
    //......
    root.searchK(K, ""); // This is the function call
                         // expected to report strings with.
}
```