**IDS Project 1**

For the ease of operation, the input is automatically set for every run in eclipse by following the steps below:

run-> run configurations->arguments-> give the required values.

**Multi-Hash Table:**

The flow of the program is as follows:

The global variables declared are:

* numTableEntries: These are the number of entries in the table
* numFlows: These are the total number of flows that we are giving
* numHashes: This is also refereed as ‘k’ which stands for the number of times that we will be hashing.
* tableVals: This is an array of values that we will be filling in our table.
* randomHashes: This is an array of random values that are used for hashing.
* Rand: this is an instance of Random class which we will be using for generating random values.

Here we have the following methods:

* Parameterized constructor -> MultiHashTable
  + This takes in the input values that are read from the command line and set them to the global variables that have been declared along with initializing the array with respective sizes.
  + For every instance of the object created here we initialize an array that is filled with random values for hashes.
* FillRandomHashes
  + This method starts off by creating a set in which we use a random number generator to fill it in with unique random values.
  + The values form this set is then iterated one by one to fill in the array of random hashes that we will be using through the program.
* hashXOR
  + Here we declare a new array and start filling it in with the hashes generated by using XOR of flow ID that is taken in and the random hashes array values that we generated earlier.
  + This generates a new hashed array whose contents are uniquely hashed.
* fillHash
  + This function helps in iterating through each flow and then assign a unique flow ID.
  + We then create a new hashedArray which is generated using the hashXOR function, the entries of this array are used to find the index of the table that we finally use, in java default value of each element in an array is 0 hence if its zero then that means it is empty hence we map or else we break out of the loop and leave that value unmapped.
* main
  + This method takes in the input and gives the values to the required variable names. We invoke the constructor by creating the object here and then we true to read the output of this from eclipse IDE console to a new file in the system.

**Cuckoo Hash Table:**

The main idea of this algorithm is to recurse through each of the segment available to put the entry in our hash table when one of the value that we are trying to hash into the table is already mapped and we try to move that entry to one of the hashed values.

The global variables declared are:

* numTableEntries: These are the number of entries in the table
* numFlows: These are the total number of flows that we are giving
* numHashes: This is also refereed as ‘k’ which stands for the number of times that we will be hashing.
* cuckooSteps: This will be determining the number of times we will be iterating for putting our values.
* tableVals: This is an array of values that we will be filling in our table.
* cMap: We use this to map the flowID’s to an array of hashedValues.
* randomHashes: This is an array of random values that are used for hashing.
* Rand: this is an instance of Random class which we will be using for generating random values.

Here we have the following methods:

* Parameterized constructor -> CuckooHashTable
  + This takes in the input values that are read from the command line and set them to the global variables that have been declared along with initializing the array with respective sizes.
  + For every instance of the object created here we initialize an array that is filled with random values for hashes.
* FillRandomHashes
  + This method starts off by creating a set in which we use a random number generator to fill it in with unique random values.
  + The values form this set is then iterated one by one to fill in the array of random hashes that we will be using through the program.
* hashXOR
  + Here we declare a new array and start filling it in with the hashes generated by using XOR of flow ID that is taken in and the random hashes array values that we generated earlier.
  + This generates a new hashed array whose contents are uniquely hashed.
* fillHash
  + This function helps in iterating through each flow and then assign a unique flow ID which is filled in the Cuckoo map.
  + We then create a new hashedArray which is generated using the hashXOR function, the entries of this array are used to find the index of the table that we finally use, in java default value of each element in an array is 0 hence if its zero then that means it is empty hence we map or else we break out of the loop and leave that value unmapped.
* recursiveSearch:
  + We check if the cuckoo steps are a positive integer, if it is then we move through the hashed array and assign a temporary index to that flow id
* main
  + This method takes in the input and gives the values to the required variable names. We invoke the constructor by creating the object here and then we true to read the output of this from eclipse IDE console to a new file in the system.

**D-left Hash Table:**

Here when we try to fill in a value in one of the hash table entries and it is filled, we move it to the immediate next available space to accommodate the one that has been mapped recently.

The flow of the program is as follows:

The global variables declared are:

* numTableEntries: These are the number of entries in the table
* numFlows: These are the total number of flows that we are giving
* numSegments: This is also refereed as ‘k’ which stands for the number of times that we will be hashing.
* tableVals: This is an array of values that we will be filling in our table.
* randomHashes: This is an array of random values that are used for hashing.
* Rand: this is an instance of Random class which we will be using for generating random values.

Here we have the following methods:

* Parameterized constructor -> DleftHashTable
  + This takes in the input values that are read from the command line and set them to the global variables that have been declared along with initializing the array with respective sizes.
  + For every instance of the object created here we initialize an array that is filled with random values for hashes.
* FillRandomHashes
  + This method starts off by creating a set in which we use a random number generator to fill it in with unique random values.
  + The values form this set is then iterated one by one to fill in the array of random hashes that we will be using through the program.
* hashXOR
  + Here we declare a new array and start filling it in with the hashes generated by using XOR of flow ID that is taken in and the random hashes array values that we generated earlier.
  + This generates a new hashed array whose contents are uniquely hashed.
* fillHash
  + This function helps in iterating through each flow and then assign a unique flow ID.
  + We then create a new hashedArray which is generated using the hashXOR function, the entries of this array are used to find the index of the table that we finally use, in java default value of each element in an array is 0 hence if its zero then that means it is empty hence, we map or else we break out of the loop and leave that value unmapped.
  + Here we multiply the index value j with 4, while iterating through inner loop so that it always falls into one of the segments that we have divided our hashes.
* main
  + This method takes in the input and gives the values to the required variable names. We invoke the constructor by creating the object here and then we true to read the output of this from eclipse IDE console to a new file in the system.