**Northeastern University**
**College of Engineering**
**Department of Electrical & Computer Engineering**

EECE7205: Fundamentals of Computer Engineering

# Spring 2020 - Homework 3

## 1. Instructions

- For programming problems:
  - Your code must be well commented by explaining what the lines of your program do. Have at least one comment for every 4 lines of code.
  - You are not allowed to use any advanced C++ library unless it is clearly allowed by the problem. For example, you cannot use a library function to sort a list of data if the problem is asking you to implement an algorithm to sort the list.
  - At the beginning of your source code files write your full name, students ID, and any special compiling/running instruction (if any).
  - Test your code on the COE Linux server before submitting it:
    a. If your program does not compile in the COE server due to incompatible text encoding format, then before uploading your source code file to the server make sure it is saved with Encoding Unicode (UTF-8). In visual studio, Save As -> Click on the arrow next to Save -> Save with Encoding -> Yes -> Unicode (UTF-8) -> Ok
    b. Compile using g++ -std=c++11 <filename>
- Submit the following to the homework assignment page on Blackboard:
  - Your homework report developed by a word processor (no handwritten or drawn contents are acceptable) and submitted as one PDF file. The report includes the following (depending on the assignment contents):
    a. Answers to the non-programming problems.
    b. A summary of your approach to solve the programming problems.
    c. A summary of the skills you acquired and challenges you faced by implementing the programs.
    d. Your recommendations of extension to the programming problems.
    e. The screen shots of the sample run(s) of your program(s)
  - Your well-commented programs source code files (i.e., the .cc or .cpp files).

  Do NOT submit any files (e.g., the PDF report file and the source code files) as a compressed (zipped) package. Rather, upload each file individually.

*Note:* Yon can submit multiple attempts for this homework, however, only your last submitted attempt will be graded.

# **Problem 1** (30 Points)

Assume we have a hash table consisting of *m* = 11 slots, and suppose nonnegative integer key values are hashed into the table using the following hash function *h1*():

```
int h1(int key) {
 int x = (key + 5) * (key + 5);
 x = x / 16;
 x = x + key;
 x = x % 11;
 return x;
}
```

The sequence of 12 integer key values listed below are to be inserted in the table, in the order given. Suppose that collisions are resolved by using linear probing. Show the probe sequence (the sequence of slots to be probed until an empty one, if any, is available) for each key. In addition, show the final contents of the hash table after all keys are inserted.

| Key Value | Probe Sequence |
|---|---|
| 43 | |
| 23 | |
| 1 | |
| 0 | |
| 15 | |
| 31 | |
| 4 | |
| 7 | |
| 11 | |
| 3 | |
| 5 | |
| 9 | |

| | Final Hash Table Contents |
|---|---|
| 0 | |
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| 10 | |

# Problem 2 (40 Points)

Write a C++ program to study how the hash table size affects the collision rate. The following are the requirements of this program:

a. Generate 1000 random integers that represent the birthdays of people who were born between January 1, 2000 and December 31, 2004 in the format *mmddyy*. An example of such integers is 112303 for someone who was born on November 23, 2003. For simplicity, assume that all people were born before the 28$^{th}$ day of their birth month and do not worry about duplicate birthdays.
b. Define four hash tables with the following sizes: $m_1$ = 97, $m_2$ = 98, $m_3$ = 100, and $m_4$ = 101.
c. Simulate storing the 1000 random birthdays that you generate in step *a* in each of the four hash tables using hash function $h(k) = k \bmod m_i$. Where $k$ is the birthday integer and $m_i$ is the size of table $i$ (i = 1, 2, 3, 4). Assume that collision is resolved by chaining. You do not need to generate the actual chains in your program instead store in each slot of the tables the number of birthdays that are mapped to that slot. Note that, at the end, if a slot of a table has the value $x$, this means there was ($x$-1) collisions on that slot.
d. For each one of the four tables, calculate the minimum, maximum, mean, and variance of the collision numbers stored in the table from step c.
e. Comments on the results you calculated in step *d* by explaining how the hash table size affects the collision rate.

# Problem 3 (30 Points)

If the root of a complete tree starts at index 1 and given the index $i$ of a node, prove the heap indices calculations as shown below:

```
PARENT(i)
1  return ⌊i/2⌋

LEFT(i)
1  return 2i

RIGHT(i)
1  return 2i + 1
```