Project 3: ESE 344 Software Techniques for Engineers, ECE, Stony Brook University, M. Subbarao

1. **Sorting and Searching** (15 points)**:** Implement the following algorithms in the Kruse and Ryba text book: can modify the code in the Kruse and Ryba text book:
   (i) **(5 pts) Quicksort algorithm**
   (ii) **(10 pts) Heap-sort algorithm**

   Test your implementation as follows:
   a. Generate 5000 integer random numbers/keys in the range 0 to 10^6 and store them in an array.
   b. Sort the array using Quicksort and Heap-sort and find the number of comparison operations on the keys/numbers in each case and print it.
   c. Repeat steps (a) and (b) above 30 times, and find the minimum, maximum, mean, median, and standard deviation of the number of comparison operations, for the two methods.
2. **Hash Tables** (10 points): Implement Hash Tables of size 8191 (= 2^13 - 1) to store integers based on (a) linear probing, and (b) quadratic probing. Choose your own hash function. Compare the average number of probes for the following cases:
   a. Generate 4000 integer random numbers in the range 0 to 10^6 - 1, and insert them into the two hash tables. Compute the number of probes for each hash table.
   b. Repeat step (a) above 10 times, and find the average number of probes over these 10 trials.

The reference book by Kruse & Ryba may be available online for downloading free. Here is a link I found (there may be others):
1. Data Structures and Algorithms , Kruse and Ryba
https://cdn.preterhuman.net/ t exts/math/Data_Structure_And_ Algorithms/Data%20Str uctures% 20and%20Program%20Design%20in% 20C++%20- %20Robert%20L.% 20Kr use.pdf

ESE 344 : Project 3 Reference sections

Text books:

1. [MW] M. A. Weiss,
2. [KR] Kruse and Ryba, Get a copy from this link:

Sorting
KR: Quicksort 8.8.1 to 8.8.3,
Heaps KR: 8.9(heap sort)
(Reference MW: 6.1 to 6.3 Heaps)

Hashing
KR : 9.6, 9.7.1 to 9.7.3 (analysis of hashing)
(Reference: MW: 5.1 to 5.3 Hashing)

```
#include <iostream>
using namespace std;
#include <cstdlib>   // for rand(), srand()
#include <ctime>     // for time()
#include <assert.h>
#include <math.h>    // for sqrt()

int main() {
	srand((unsigned int) time(NULL));// seed rand() with system time
	for (int i = 0; i<100; i++) {
			cout<< (rand() % 100) <<endl; // limit data to 0 to 99
	}

// Examples of hashing a 6 character long string and 6 digit long integer are
given
// given below. However, you have to change the hash functions in your
project.
// Your hash functions should be complicated and creative enough to generate
random
// outputs. You will lose points if you make only trivial changes
// to these functions.



#include <iostream>
#include <vector>
#include <string>
#include <math.h>
#include <cassert>
#include <cstdlib>   // for rand(), srand()
#include <ctime>     // for time()

using namespace std;

// Prof. Murali Subbarao, ESE 344, March 2023

// An example of hashing a 6 character long sting

int hashStr1(string s = "abcdef") {
	long int h=0;
	for (int i = 0; i < s.length(); i++) {
		h += ( ((unsigned)s[i]) * ((unsigned)s[i]) ) ;
	}

	h = (h % 997);
	return (int) h;
}
```

```
// An example of hashing a 6 digit integer

int hashInt1(int n = 734906) {
        int m, k, h;
        k = 100; h = 0;
        while (n > 0) {
                m = n % k;
                h += (m * m + 7);
                n = n / k;
        }
        h = (h % 997);
        return h;
}
int main()
{
        using namespace std;
        int n1, n2, n3, n4;
        char c;


        srand((unsigned int)time(NULL));// seed rand() with system time

        cout << "Enter a 6 digit integer for hashing, e.g. 734906: "  << endl;
        cin >> n1;
        if ((n1 < 1) || (n1 > 1000000)) {
                cout << "n1 is out of range . " << endl;
                return 1;
        }

        cout << "n1 : " << n1 << endl;
        cout << hex << "n1 in hex : " << n1 << endl;
        cout << dec << "n1 in dec : " << n1 << endl;
        n2 = hashInt1(n1);

     cout << "n2 : " << n2 << endl;

        cout << endl << "Enter any char to continue : ";
        cin >> c;

        string s1;
        cout << "Enter a 6 character string for hashing, e.g. abcdef : " <<
endl;
        cin >> s1;
        if (s1.length() !=6) {
                cout << "s1 length is not 6 . " << endl;
                return 1;
        }
        cout << "s1 : " << s1 << endl;
        for (int i = 0; i < s1.length(); i++) {
                cout << "  s1[" << i << "] : " << s1[i];
        }
        cout << endl;
        for (int i = 0; i < s1.length(); i++) {
                cout << "  s1[" << i << "] : " << (unsigned) s1[i];
        }
        cout << endl;
        for (int i = 0; i < s1.length(); i++) {
            cout <<dec << "  s1[" << i << "] : " << hex<<(unsigned)s1[i];
        }
        cout << dec <<endl;
        n3 = hashStr1(s1);

        cout << "n3 : " << n3 << endl;

        cout << endl << "Enter any char to continue : ";
        cin >> c;


        return 0;
}
```

An example of Test Input:
587913
c
Ese344
C

Output for the test input above:

```
Enter a 6 digit integrer for hashing, e.g. 734906:
n1 : 587903
n1 in hex : 8f87f
n1 in dec : 587903
n2 : 662

Enter any char to continue : Enter a 6 character string for hashing, e.g. abcdef :
s1 : pkrtzn
  s1[0] : p  s1[1] : k  s1[2] : r  s1[3] : t  s1[4] : z  s1[5] : n
  s1[0] : 112  s1[1] : 107  s1[2] : 114  s1[3] : 116  s1[4] : 122  s1[5] : 110
  s1[0] : 70  s1[1] : 6b  s1[2] : 72  s1[3] : 74  s1[4] : 7a  s1[5] : 6e
n3 : 660

Enter any char to continue :
```