Hash Tables

Exercise 1

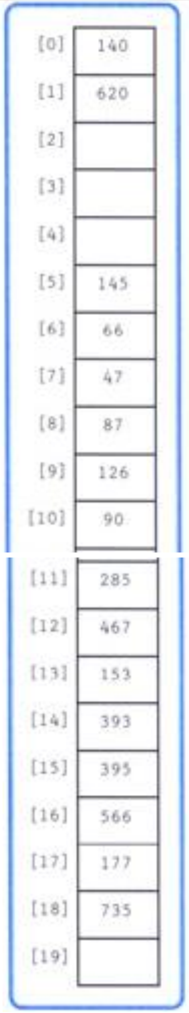
- Find a hash function to convert numeric personal numbers into values between 1 and 10. Write a program to generate some random numeric personal numbers test your function.

Exercise 2

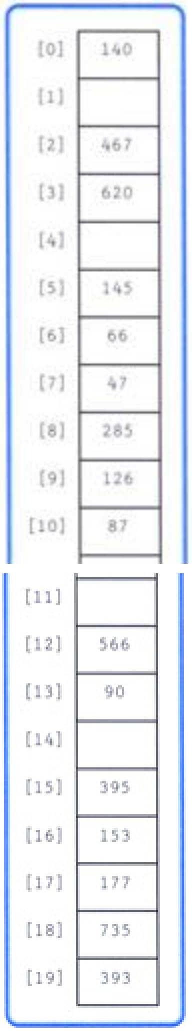
- Use the following values:
- *66 47 87 90 126 140 145 153 177 285 393 395 467 566 620 735*
- Store the values into a hash table with 20 positions, using the division method of hashing and the linear probing method of resolving collisions.

Hints:

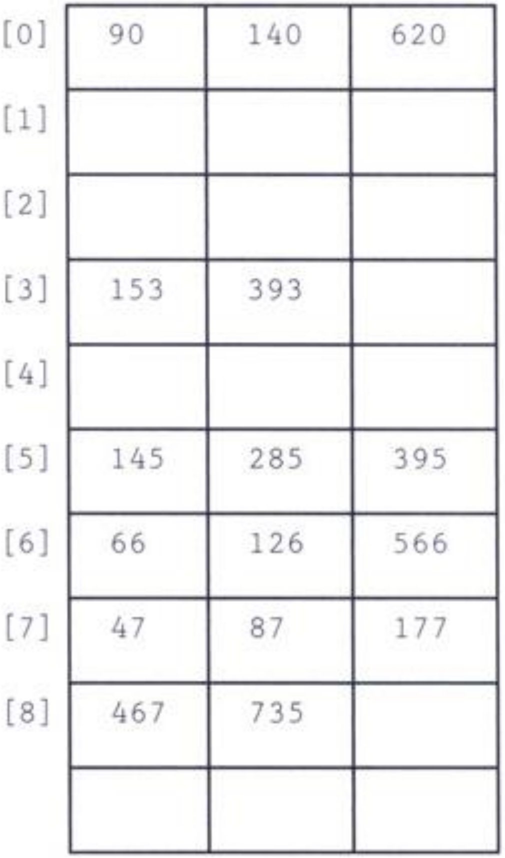
- Store the values into a hash table with 20 positions, using rehashing as the method of collision resolution. Use key % tableSize as the hash function, and (key + 3) % tableSize as the rehash function.
- Store the values into a hash table with ten buckets, each containing three slots. If a bucket is full, use the next (sequential) bucket that contains a free slot.
- Store the values into a hash table that uses the hash function key % 10 to determine into which of ten chains to put the value.



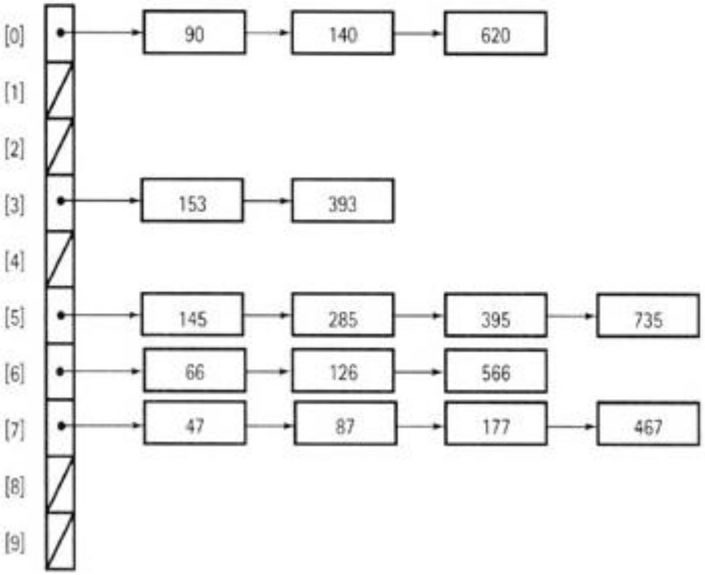
[0]	140
[1]	620
[2]	
[3]	
[4]	
[5]	145
[6]	66
[7]	47
[8]	87
[9]	126
[10]	90
[11]	285
[12]	467
[13]	153
[14]	393
[15]	395
[16]	566
[17]	177
[18]	735
[19]	



[0]	140
[1]	
[2]	467
[3]	620
[4]	
[5]	145
[6]	66
[7]	47
[8]	285
[9]	126
[10]	87
[11]	
[12]	566
[13]	90
[14]	
[15]	395
[16]	153
[17]	177
[18]	735
[19]	393



[0]	90	140	620
[1]			
[2]			
[3]	153	393	
[4]			
[5]	145	285	395
[6]	66	126	566
[7]	47	87	177
[8]	467	735	



Hash Table Implementation

```
#include <stdio.h>
#include <string.h>
#include "linked_list.h"
#define VMAX 17
#define P 13

template<typename Tkey, typename Tvalue> struct elem_info {
    Tkey key;
    Tvalue value; };

template<typename Tkey, typename Tvalue> class Hashtable {
    private:
        LinkedList<struct elem_info<Tkey, Tvalue> > *H;
        int HMAX;
        int (*hash) (Tkey);

        public:
            Hashtable(int hmax, int (*h) (Tkey)) {
                HMAX = hmax;
                hash = h;
                H = new LinkedList<struct elem_info<Tkey,
                                                                  Tvalue> > [HMAX]; }

            ~Hashtable() {
                for (int i = 0; i < HMAX; i++) {
                    while (!H[i].isEmpty())
                        H[i].removeFirst();
                }

                delete H;
            }
```

Part 2

```
void put(Tkey key, Tvalue value) {
                struct list_elem<struct elem_info<Tkey, Tvalue> > *p;
                struct elem_info<Tkey, Tvalue> info;

                int hkey = hash(key);
                p = H[hkey].pfirst;

                while (p != NULL) {

              if (p->info.key == key)
                break;
                p = p->next;
              }

              if (p != NULL)
                p->info.value = value;
              else {
                info.key = key;
                info.value = value;
                H[hkey].addLast(info);
              }
            }






                                  };
```

```
Tvalue get(Tkey key) {
      struct list_elem<struct elem_info<Tkey, Tvalue> > *p;

      int hkey = hash(key);
      p = H[hkey].pfirst;

      while (p != NULL) {
        if (p->info.key == key) break;
        p = p->next;
      }

      if (p != NULL)
        return p->info.value;
      else {
        fprintf(stderr, "Error 101 - The key does not exist in the hashtable\n");
        Tvalue x;
        return x;
      }
    }

int hasKey(Tkey key) {
      struct list_elem<struct elem_info<Tkey, Tvalue> > *p;

      int hkey = hash(key);
      p = H[hkey].pfirst;

      while (p != NULL) {
        if (p->info.key == key)
          break;
        p = p->next;
      }

      if (p != NULL)
        return 1;
      else
        return 0;
    }
```

Part 3

```
int hfunc(int key) {
    return (P * key) % VMAX;
}

Hashtable<int, double> hid(VMAX, hfunc);

int hfunc2(char* key) {
    int hkey = 0;
    for (int i = 0; i < strlen(key); i++)
        hkey = (hkey * P + key[i]) % VMAX;
    return hkey;
}

Hashtable<char*, int> hci(VMAX, hfunc2);

char *k1 = "abc";
char *k2 = "xyze";

char *k3 = "Abc";
char *k4 = "abcD";

int main() {
    hid.put(3, 7.9);
    hid.put(2, 8.3);
    printf("%.3lf\n", hid.get(3));
    hid.put(3, 10.2);
    printf("%.3lf\n", hid.get(3));
    printf("%.3lf\n", hid.get(2));
    printf("%d\n", hid.hasKey(5));
    printf("%d\n", hid.hasKey(2));
    printf("%.3lf\n", hid.get(5));

    hci.put(k1, 10);
    hci.put(k2, 20);
    printf("%d\n", hci.get(k1));
    hci.put(k1, 30);
    printf("%d\n", hci.get(k1));
    printf("%d\n", hci.get(k2));
    printf("%d\n", hci.hasKey(k3));
    printf("%d\n", hci.hasKey(k2));
    printf("%d\n", hci.get(k4));

    return 0;
}
```