Lab 5

Due No Due Date **Points** 1

You are suggested to use the teaching servers burrow.soic.indiana.edu or hulk.soic.indiana.edu or tank.soic.indiana.edu for practicing C programs.

Lab 5: Bucket sort with linked list

For this lab, all students must have a good grasp on linked list.

Random number generation

The following function generates and assign random numbers within the range (0,1) to the array with n elements

```
#include <stdlib.h>

void generateRandomNumbers(double a[], int n)
{
        int i;
        for(i = 0; i < n; ++i)
        {
                // rand() generates a random number from 0 to RAND_MAX, defined in <stdlib.h>
                // After the division with RAND_MAX, the array will have random value within (0,1)
                a[i] = rand() / (double)RAND_MAX;
        }
}
```

Linked List

We will use one way linked list in this exercise with a floating point value and a pointer to the next node of the linked list. A generic node can be created this way:

```
typedef struct node_struct
{
        double d;
        struct node_struct *next;
} node;
```

An array of node pointers can be created and initialized with NULL the following way which can be used in a bucker sort.

```
node *B[100];
for(i = 0; i < n; ++i)
        B[i] = NULL;
```

Now we have n node pointers that we can use for creating n separated linked lists.
The following function takes a value and inserts it into the index-th linked list.

```c
void insert(double value, int index, node *B[])
{
        // This function insert a new node with value into the B[index] linked list. The function
        // inserts the new node in the correct place of the linked list so that the link list is sorted

        node *t;
        if (B[index] == NULL)
        {
                // No elements in the linked list, create and insert the node at the beginning of the list
                t = (node *)malloc(sizeof(node));
                t->d = value;
                t->next = NULL;
                B[index] = t;
        }
        else
        {
                // Take two pointers p0 and p1. p0 always stays one node behind p1
                // The new node t will be inserted either on the end of the linked list
                // or before a node that has a value greater than the new node value.

                node *p0, *p1;
                p0 = B[index];
                p1 = p0->next;

                if (p0->d > value)
                {
                        t = (node *)malloc(sizeof(node));
                        t->d = value;
                        t->next = p0;
                        B[index] = t;
                }
                else
                {
                        while (p1 != NULL)
                        {
                                if (p1->d > value)
                                {
                                        break;
                                }
                        // Advance both node pointer one node ahead to compare with next element in the lin
ked list
                                p1 = p1->next;
                                p0 = p0->next;
                        }
                // Create new node t and insert at the appropriate place
                        t = (node *)malloc(sizeof(node));
                        t->d = value;
                        t->next = p1;
                        p0->next = t;
                }
        }
}
```

Bucket Sort

Bucket sort sorts number drawn from uniform distribution in average O(n) time. Bucket sort divides the input numbers (0,1) into n equal sized separate slot (i.e a linked list) and maps the values to a slot and insert the value into that slot (i.e linked list). Each linked list maintain a sorted order by inserting all values with insertion sort.

NOTE: For this assignment, you need to implement the data structure linked list yourself, using already existed implementation is NOT accepted.

```
BUCKET-SORT(A)
        let B[0...n-1] be a new array
        n = A.length
        for i = 0 to n-1
                make B[i] an empty list
        for i = 1 to n
                insert A[i] into the slot B[index] in sorted order using insertion sort, where index=floor
(n*A[i])
        print out the list values of B[0...n-1] to output the sorted numbers
```

Test

Generate uniformly distributed random array of size 10,100,1000 etc. and test your algorithm on them.