COP-4338: Systems Programming

Programming Assignment 4
PART II

In this assignment, you are asked to write a program that gets a series of string chars followed by EOF from user through console, alphabetically sorts them using bucket sort, and prints the sorted strings using printf function.

1 Bucket Sort

Bucket sort, or bin sort, is a sorting algorithm that works by distributing the elements of a list into a number of buckets. Each bucket is then sorted individually and then the sorted list is obtained by merging all the sorted buckets together. To sort each bucket in this assignment, you need to use any algorithm that you want. If you have no preference, you can use *bubble sort* algorithm which is fairly easy to implement. Also, to compare strings with one another, you need to use function *strcmp*.

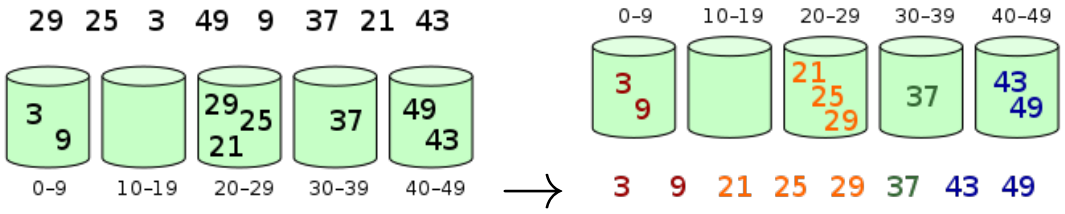

Figure 1: Example of bucket sort for sorting numbers 29, 25, 3, 49, 9, 37, 21, 43. Numbers are distributed among bins (left image). Then, elements are sorted within each bin (right image).

2 Buckets of Your Algorithm

In order to implement bucket sort algorithm for strings, you need to have an array of buckets such that *bucket* is a C structure defined in the following way:

```
struct bucket{
    int minInitial;
    int maxInitial;
    int numWords;
    struct node* chainHead;
};
```

and *node* is another C structure defined in the following way:

```
struct node{
    char* string;
    struct node* next;
};
```

The array of buckets must be defined and initialized in the following way:

```
struct bucket list[NUMBUCKS] = {
    'a', 'b', 0, NULL,
    'c', 'c', 0, NULL,
    'd', 'f', 0, NULL,
    'g', 'k', 0, NULL,
    'l', 'o', 0, NULL,
    'p', 'r', 0, NULL,
    's', 's', 0, NULL,
    't', 'z', 0, NULL
};
```

where NUMBUCKS represents 8. Therefore, you have eight buckets in your program. Each bucket contains four variables:

- int minInitial: determines the minimum ASCII code of the initial letter which the strings in the bucket can start with.
- int maxInitial: determines the maximum ASCII code of the initial letter which the strings in the bucket can start with.
- int numWords: determines the number of strings placed in the bucket.
- struct node* chainHead: points to a singly linked list which is a chain of nodes containing the strings in the bucket.

3 Program Input

Assume that user enters a series of string chars followed by an EOF. Also, assume that every input string *only* contains *letters* of English Alphabet in *lowercase* format. Every string of chars that user passes to the program must be stored in the format of a char string. To allocate memory for storing a char string in memory, you can use expression *char* aWord = (char *)malloc(strlen(inputString)+1);* in which *inputString* is a *char ** pointing to the array of letters given by the user, *strlen* and *malloc* are functions defined in *stdlib.h* library (note that *+1* in the mentioned expression allocates an extra memory cell for \0 that specifies the end of a char string).

After allocating enough memory cells for each input string and copying the string into them, you need to allocate memory for storing a variable of type *struct node* containing a *char ** (pointing to the string) and a *struct node ** pointing to the next node in the linked list of appropriate bucket.

4 Program Output

Your program must print out the sorted list of strings in an alphabetical order using *printf* function.

5 Submissions

You need to submit a *.zip* file compressing the C source file(s) related to the assignment (*.c* files) along with a readme.txt file explaining which sorting algorithm you used for sorting each bucket and listing parts of the assignment has been implemented completely/partially.