THE UNIVERSITY OF WESTERN ONTARIO

DEPARTMENT OF COMPUTER SCIENCE
LONDON CANADA

Software Tools and Systems Programming
(Computer Science 2211a)

ASSIGNMENT 5
Due date: Friday, November 30, 2018, 11:55 PM

Assignment overview

Objectives. The purpose of this assignment is to get experience with

- advanced data structures and ADT,
- manipulation of C pointers and C structures,
- dynamic allocation and and deallocation of memory,
- organizing code in multiple files,
- writing Makefile to compile your code.

In this assignment, you are to write a C program to implement an associative matrix structure (2D array) indexed by a pair of strings using binary search trees.

Assignment basic requirements. The code should be well documented and logically organized. The comments should be proper. Your code should be tested carefully before submitting it, especially for boundary cases, such as empty data-structures. Avoid segmentation fault and memory leak.

1 Preliminaries

In this assignment, you will implement the following data structures.

Binary search tree

This will be implemented with pointers and structures. The key type is a pair of strings (a pair of pointers to char) which will be used later as indices of the matrix structure. The data type can be any type and in this assignment it is pointer to int.

The type definitions for key and data in C are the following.

```
typedef int* Data;
typedef struct {char *skey1; char *skey2;} Key_struct;
typedef Key_struct* Key;
```

You will need a function to duplicate a string, a function to generate a key from a pair of strings, a function to compare two keys, a function to print a key, a function to free memory used by a key, a function to generate a data from an integer, a function to set a data from an integer, a function to print a data, and a function to free memory used by a data.

`char * string_dup(char *str);`
Duplicate string pointed to by **str** with dynamic memory allocation.

`Key key_gen(char *skey1, char *skey2);`
Generate a key with dynamic memory allocation.

`int key_comp(Key key1, Key key2);`
Use strcmp( ) to do comparison. If **key1->skey1** and **key2->skey1** are different, then they determine the comparison result. If **key1->skey1** and **key2->skey1** are the same, then **key1->skey2** and **key2->skey2** determine the comparison result.

`void key_print(Key key);`
Print a key.

`void key_free(Key key);`
Free memory allocated for key.

`Data data_gen(int idata);`
Generate a data with dynamic memory allocation.

`void data_set(Data data, int idata);`
Assign data with idata.

`void data_print(Data data);`
Print a data.

`void data_free(Data data);`
Free memory allocated for data.

The type definitions for binary search trees are the following:

```
typedef struct BStree_node {
  Key key;
  Data data;
  struct BStree_node *left, *right;
} BStree_node;
typedef BStree_node** BStree;
```

The operations for binary search trees are the following.

`BStree bstree_ini(void);`
Allocate memory of type BStree_node*, set the value to NULL, and return a pointer to the allocated memory.

`void bstree_insert(BStree bst, Key key, Data data);`
Insert **data** with **key** into **bst**. If **key** is in **bst**, then do nothing.
You may want to use a helper function for insertion to create a pointer to a tree node from key and data.

```
BStree_node *new_node(Key key, Data data);
```

`Data bstree_search(BStree bst, Key key);`
If **key** is in **bst**, return **key**'s associated data. If **key** is not in **bst**, return NULL.

`void bstree_traversal(BStree bst);`
In order traversal of **bst** and print each node's key and data.

`void bstree_free(BStree bst);`
Free all the dynamically allocated memory of **bst**.

A Matrix Indexed by a pair of Strings
The matrix structure will be implemented as Matrix using BStree.

The type definition in C is the following.

```
typedef BStree Matrix;
typedef char* Index;
typedef int Value;
```

The operations are the following.

`Matrix matrix_construction(void);`
Matrix construction using bstree_ini();

`unsigned char matrix_isin(Matrix m, Index index1, Index index2);`
If location (**index1, index2**) is defined in Matrix **m**, then return 1. Otherwise, return 0.

`Value *matrix_get(Matrix m, Index index1, Index index2);`
If location (**index1, index2**) is defined in Matrix **m**, then return a pointer to the associated value. Otherwise, return NULL.

`void matrix_set(Matrix m, Index index1, Index index2, Value value);`
Assign **value** to Matrix **m** at location (**index1, index2**). If that location already has value, then overwrite.

`void matrix_inc(Matrix m, Index index1, Index index2, Value value);`
If location (**index1, index2**) is defined in Matrix **m**, then increase the associated value by **value**. Otherwise, report error.

`void matrix_list(Matrix m);`
Print indices and values in the Matrix **m** (with bstree_traversal()).

`void matrix_destruction(Matrix m);`
Free allocated space (with bstree_free()).

When implementing BStree and Matrix, you are free to use helper functions.

2 Organizing the code into multiple files
For this assignment you are to organize the code in the following way:

- In the file datatype.h, define the type Data, the type Key, and declare prototypes of the functions for type Data and type Key.
- In the file datatype.c, implement the functions for type Data and type Key.
- In the file bstree.h, define the type BStree_node, the type BStree and declare prototypes of the operations on BStree.
- In the file bstree.c, implement the functions on BStree.
- In the file matrix.h, define the type Index, the type Value, and the type Matrix and declare prototypes of the operations on Matrix.
- In the file matrix.c, implement the functions on Matrix.
- In the file main.c, your program will
    1. create a new Matrix.
    2. read from stdin, or redirect from a file, string pairs (a pair of strings, i.e. two strings, per line) and then calculate occurrences of each string pair read using the Matrix created.
    3. print the data in the Matrix
    4. free all allocated memory spaces for the Matrix and terminate.

A sample input is given below.

```
bba    aa
 aab  aab
bba  aa
aab  abb
bba   aaa
```

A sample output is given below.

```
String 1  String 2   Occurrence
aab       aab        1
aab       abb        1
bba       aa         2
bba       aaa        1
```

3 Creating a Makefile to compile the source code

You are asked to create a Makefile to compile your source code. When "make" is typed, an executable program called "mymatrix" is generated. Typing "make clean" delete all the files generated by "gcc".

4 Testing your program

You should first implement functions related to type Key and Data. Test these functions to make sure they are correct.
Then implement BStree and test it to make sure it is correct before implementing Matrix.
Your program should have no segmentation fault and no memory leak. Your program should print all the elements correctly.
You should test your program by running it on Gaul. Capture the screen of your testing by using script command.
You should submit your assignment through OWL. Please check the assignment submission guidelines.