# Assignment #5

## Computer Science I – COP 3502

**Objectives**

1. To learn how to use binary search trees
2. To practice implementing known algorithms

**Problem: Party Games (games.c)**

This program is designed to organize a set of people and their favorite childhood games. This is split into two parts.

In part one, we will organize people based simply on names. The program must accept simple commands (read from a file) to add people, remove people, print a listing of everyone, and print a list of people who have a particular favorite game.

In part two, we will organize people first based on their favorite game and then alphabetically. Then, we will be able to print all the games with a listing of the people who like to play them.

**Program Instructions – Part 1**

In part one, you must use a Binary Search Tree to store the information. The ordering property should be based on the names of the people as determined by the prewritten function strcmp. You are guaranteed that all names will be unique.

The program will follow an input file titled "party.txt", which will contain several commands:

- ADD <name> <game>
  - This command should add a new person to the BST with the indicated data
- REMOVE <name>
  - This command should remove the person with the indicated name from the BST
- PRINTGAME <game>
  - This command should print the names of all the people whose favorite game is indicated by game, using an inorder traversal.
- PRINT
  - This command should print the names of all the people in the BST with their favorite game, using an inorder traversal.
- PART2
  - This command should begin part two of the program.

Please view the output specification below for precise output.

**Program Instructions – Part 2**

In part two, you must use a dynamically allocated array to store the same information from part one. Then, implement the MERGESORT sorting algorithm to sort the array, first based on the favorite game and then based on the name of the person (as determined by the prewritten function strcmp).

Once the array is sorted, simply print the contents of the array as shown in the output specification.

**Recommendations**

The following structure is recommended, but not required, for your program.

```
struct person {
   char name[20];
   char game[20];
};
```

**Specifications**

These specifications must be followed exactly.

**Input:**

1. The input location will be an input file titled “party.txt”.
2. All commands, names, and games will be UPPERCASE
3. The file will contain commands for part 1, with one command per line
    a. ADD <name> <game>
    b. REMOVE <name>
    c. PRINTGAME <game>
    d. PRINT
4. The last line of the file will be PART2, indicating that part 1 is complete

**Sample Input File:**

```
ADD KAREN MONOPOLY
ADD MARK SCRABBLE
ADD JOHN MONOPOLY
ADD SASHA WAR
ADD COURTNEY SCRABBLE
PRINT
REMOVE SASHA
ADD DAVID MONOPOLY
PRINTGAME MONOPOLY
PART2
```

**Output:**

1. Output location must be an output file titled “partygames.out”
2. The following commands produce output. Use the indicated format to complete the command.
    a. PRINT
        i. PRINT:
        ii. <name> <game>
    b. PRINTGAME
        i. <game>:
        ii. <name>
        iii. <name>
        iv. …
    c. PART2
        i. ---Part 2---
        ii. <game>
            1. <name>
            2. <name>
            3. …

**Sample Output File:**

PRINT:
COURTNEY SCRABBLE
JOHN MONOPOLY
KAREN MONOPOLY
MARK SCRABBLE
SASHA WAR

MONOPOLY:
DAVID
JOHN
KAREN

---PART 2---
MONOPOLY
    DAVID
    JOHN
    KAREN
SCRABBLE
    COURTNEY
    MARK

**Program:**

1. You must use a BST for part one
2. You must use an array of structs for part two
3. You must write separate functions for each of the following tasks
    a. Add element to the BST
    b. Remove element from the BST
    c. Print all elements of the BST
    d. Print specific elements of the BST
    e. mergesort

**Deliverables**

One source file – *games.c* – is to be submitted over WebCourses.

**Restrictions**

Your program must be able to be compiled and executed in a standard C Development Environment. Yourprogram should include a header comment with the following information: your name, course number, section number, assignment title, and date. Also, make sure youinclude comments throughout your code describing the major steps in solving the problem.

**Grading Details**

Your programs will be graded upon the following criteria:

1) Your correctness

2) Your programming styleand use of white space. Even if you have a plan and your program works perfectly, if your programming style is poor or your use of white space is poor, you could get 10% or 15% deducted from your grade.

3) Your comments. Each function must have appropriate pre and post conditions. Internal comments should be used liberally – both in main and in any functions.

4) Compatibility – You must submit C source files that can be compiled and executed in a standard C Development Environment. If your program does not compile, you will get a sizable deduction from your grade.