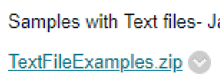
Professor C's Help and Hints: Program Set 3

- Read and go through the problems. Look at the sample runs I provide with each problem to help you with input and output understanding. Trace by hand if needed. Practice the skills you learned in previous programming courses.
- Make sure you are selecting the correct number of problems from the appropriate sections.
- These problems should be solved with Linked Lists, Recursion, and Sort/Searching. Use your past knowledge from COSC 1436/COSC 1437 as well. Remember you build from what you learned previously so everything up to this point is fair game to use. You may NOT use:
    - Built in Sort or Search functions/methods found in Java/C++ class libraries (i.e. `sort()`, etc.)
- I gave you all examples in both Java and C++ on how to read in a data file from keyboard. Please look at those examples. Under the PS 3 Assignment.



Samples with Text files- Java and C++

TextFileExamples.zip

- Remember I will test your programs with other data set than those examples given in the problem. If your program meets all the input requirements, they should work correctly for those as well. The additional judging data/files will not be provided to you.
- It is hard for me to post actual code because I do not always have solutions in both languages. Also, I may have solutions from previous semesters, but the problem may have been modified so it is not problem now as it was previously. Keep in mind at this level you should be able to put things together and modify code. My help with code is limited to what I said in the Course Agreement posted at the start of the semester. There is nothing wrong getting help from other sources (tutors, relatives, friends, colleagues, etc.) as long as in your comment block you state where you got help from.

`RosePetals`

- Use a `LinkList` or `List`
- This is like the Josephus Problem:
  https://www.geeksforgeeks.org/josephus-problem-set-1-a-on-solution/
- So with some modification we could find the loser as:

```
int loser = (numPetals - 1) % a.size();
```

`StableMarriage`

- Use the following general algorithm:

```
set each person to be free;
while (some man m with a nonempty preference list is free) {
    w = first woman on m's list;
    if (some man p is engaged to w) {
        set p to be free
    }
    set m and w to be engaged to each other
    for (each successor q of m on w's list) {
        delete w from q's preference list
        delete q from w's preference list
    }
}
```

BlobGenerator

- Here is some help in Java with filling the random values into the 2D array:

```
int randInt;
    Random rand = new Random();

    for (int i = 0; i < rows; i++)
      for (int j = 0; j < cols; j++)
      {
        randInt = rand.nextInt(100);  // random number 0 .. 99
        if (randInt < percentage)
          grid[i][j] = true;
        else
          grid[i][j] = false;
      }
```

- Here is some help in Java with recursively finding the blob

```
  void markBlob(int row, int col)
  // Mark position row, col as having been visited.
  // Check and if appropriate mark locations above, below, left,
  // and right of that position.
  {
    visited[row][col] = true;

    // check above
    if ((row - 1) >= 0)           // if its on the grid
      if (grid[row - 1][col])       // and has a blob character
        if (!visited[row - 1][col])   // and has not been visited
          markBlob(row - 1, col);       // then mark it

    // check below
    if ((row + 1) < rows)        // if its on the grid
      if (grid[row + 1][col])       // and has a blob character
        if (!visited[row + 1][col])   // and has not been visited
          markBlob(row + 1, col);       // then mark it

    // check left
    if ((col - 1) >= 0)           // if its on the grid
      if (grid[row][col - 1])       // and has a blob character
        if (!visited[row][col - 1])   // and has not been visited
          markBlob(row, col - 1);       // then mark it
    // check right
    if ((col + 1) < cols)        // if its on the grid
      if (grid[row][col + 1])       // and has a blob character
        if (!visited[row][col + 1])   // and has not been visited
          markBlob(row, col + 1);       // then mark it
  }
}
```

SchoolRankReport

- Here is some quazi/code help with the calculation of the points:

```
percent = (double)corr / (corr + incor);
score = corr * 6 - incor * 2;
```

CrewScheduling

- Notice this time you are writing the output to a datafile not to a screen. Should be some help with this under the `TextFileExamples` from above.
- The user must enter the datafile to be read from the keyboard. The output file name may or may not be specified by the user from the keyboard.
- Here is a C/C++ base way a previous student set up a similar problem without linked data.

```
#define NUM_CREWS 17
#define NUM_WEEKS 17
#define MAX_WEEKLY_GAMES 16
#define NUM_TEAMS 32
#define MAX_BYES 2
#define MIN_SEPARATION 6

enum TeamSide
{
	home,
	away
};

// Crew class declaration
class Crew;

// Function declarations
int findTeam(string name);
string extractTeam(string game, TeamSide side);
void findPlayableGames(int, bool[NUM_CREWS][NUM_CREWS], int[NUM_CREWS],
int[NUM_CREWS]);
bool scheduleWeek(bool[NUM_CREWS][NUM_CREWS], int[NUM_CREWS], int[NUM_CREWS],
bool[NUM_CREWS], bool[NUM_CREWS], int[NUM_CREWS]);
bool scheduleCrews();
bool loadData(ifstream& in);

// Global variables
unsigned int longestName = 0;
```

```
string teams[NUM_TEAMS];
Crew* crews[NUM_CREWS];
string weeklyGames[NUM_WEEKS][MAX_WEEKLY_GAMES];
```

- Here is another segment in Java from a previous student creating a user interface.

```
System.out.print("Enter a file name for the games: ");
Scanner inputG = new Scanner(System.in);
gamesFile = inputG.nextLine();

CrewSchedulingII nfl = new CrewSchedulingII();
File file = new File("Schedule.txt");
  file.delete();
   file=null;

String weekOrCrew = "w";
System.out.println("Do you want to see the schedule by week(w) or by crew(c)?");
                   weekOrCrew = sys.next();

      if (weekOrCrew.charAt(0) == 'c' || weekOrCrew.charAt(0) == 'C') {
System.out.println("Do you want the schedule for all crews (a) or one crew (Rxx)?");
                          allOrOneCrew = sys.next().toUpperCase();
             if (allOrOneCrew.charAt(0) != 'R' && allOrOneCrew.charAt(0) != 'A') {
                                 System.out.println("Invalid Data Entry " +
allOrOneCrew + ". All crews will be printed.\n");
                                 allOrOneCrew = "A";
                          }
                          System.out.println("Building Crew Schedule I");
                          nfl.run();
                          nfl.printCrewSchedule("Schedule I", allOrOneCrew);

                          System.out.println("\nBuilding Crew Schedule II");
                          nfl.run();
                          nfl.printCrewSchedule("Schedule II", allOrOneCrew);
                   }
                   else {
                          System.out.println("\nBuilding Weekly Schedule");
                          nfl.run();
                          nfl.printGameSchedule();

                   }
System.out.println("\nSchedule complete.  File is called Schedule.txt");
      }
```

Extra Credit

`SortDistance`

`SearchKey`

- Both problems above should be straightforward.