Group Project List

These projects are meant to develop your understanding of the basic principles of Data Structures and Algorithms. You will work in pre-selected groups of three to four students. Students must work in groups—no individual projects will be accepted. You and your team members will evaluate group participation at the project's conclusion. Your group will receive a single project mark, but your individual course mark can be affected by the results of your team members' evaluations.

It is possible that most teams will solve the same project problem. It must be noted that marks will be awarded based on the comparative quality of work presented. (ie. how the solution presented relates to others submitted by other groups. This means that there is no way a group can score more or equal to the best solution in each category.)

Deliverables:

Every group will prepare a 10min presentation to be delivered before the class to demonstrate:

1. The definition of the problem
2. Procedure in solving the problem (Algorithm development, choice of data structures, etc)
3. Implementation of Solution (Running of the written programme, etc)
4. Any suggestions for future improvements where necessary

In addition to the above, every group must hand in a short report not be more than four pages prior to their presentation. The report should include the names of every group member and their contribution to the group work. This must be signed by every member to confirm the validity of the content. In addition to these, the contents of the report should follow the same format as that of the presentation.

Marks:

This project contributes 50% towards your coursework. The mid semester exam takes the other 50%. The mark distribution will be as follows:

1. Presentation
    a. Problem definition (Demonstration of clear understanding of the problem): **10%**
    b. Steps taken to address the problem (Group must justify their approach (algorithm) and explain their method of solving the problem.) : 3**0%**
    c. Implementation of Solution – The group must demonstrate the implementation of their solution through a sample program and validate it: **40%**
    d. Quality of the implementation (This will be in comparison to other solutions given): **20%**

Deadline: 28th March, 2024

Projects will be pre-assigned to groups.

Project List:

1. Write a simple airline ticket reservation program. The program should display a menu with the following options: reserve a ticket, cancel a reservation, check whether a ticket is reserved for a particular person, and display the passengers. The information is maintained on an alphabetized linked list of names. To simplify the task, assume that tickets are reserved for only one flight.

2.

a. Write a program to convert a number from decimal notation to a number expressed in a number system whose base (radix) is a number between 2 and 9. The conversion is performed by repetitious divistion by the base to which a number is being converted and then taking the remainders of the division in the reverse order. For example, in converting to binary, the number 6 requires three such divisions: 6/2 = 3 remainder 0, 3/2 = 1 remainder 1, and finally, 1⁄2 = 0 remainder 1. The remainders 0, 1 and 1 are put in the reverse order so that the binary equivalent of 6 is equal to 110.

b. Modify the program so that it can perform a conversion in the case when the base is a number between 11 and 27. Number systems with bases greater than 10 require more symbols. Therefore, use capital letters. For example, a hexadecimal system requires 16 digits: 0, 1.....A, B, C, D, E, F. In this system, decimal number 26 is equal to 1A.

3. Write a simple dictionary programme that enables a user to find the definition of a word. It must permit addition, amendment, deletion, etc of words and possibly suggest other words of similar meanings during search.
4. A common problem for compilers and text editors is to determine if the parentheses (or other brackets) in a string are balanced and properly nested. For example, the string "((())())()" contains properly nested pairs of parentheses, but the string ")()(" does not, and the string "())" does not contain properly matching parentheses.

a. Give an algorithm that returns **true** if a string contains properly nested and balanced parentheses, and **false** otherwise. Use a stack to keep track of the number of left parentheses seen so far. **Hint:** At no time while scanning a legal string from left to right will you have encountered more right parentheses than left parentheses.
b. Give an algorithm that returns the position in the string of the first offending parenthesis if the string is not properly nested and balanced. That is, if an excess right parenthesis is found, return its position; if there are too many left parentheses, return the position of the first excess left parenthesis. Use a stack to keep track of the number and positions of the left parentheses seen so far.

c. Implement the above (a and b) as member functions of a class and test them by using them in a sample program.

It will be advisable to implement this project using classes and then use these objects in your main programme.

5. A **palindrome** is a string that reads the same forwards as backwards. For example, "refer". Write an algorithm to determine if a string is a palindrome. Assume that the string is read from standard input one character at a time. The algorithm should output **true** or **false** as appropriate. Consider using multiple stacks.
    a. Implement your algorithm in the form of a computer program to validate it.
6. Write a program that reads a text file and outputs the 100 most frequent words in the text. You have to determine the appropriate data structure to make this task efficient and then develop an algorithm to solve the problem.