CSE102 – Computer Programming (Spring 2021)

Homework # 11

Handed out: May 31, 2021.

Due: 11.55pm June 15, 2021.

Hand-in Policy: Via Teams. No late submissions will be accepted.
Collaboration Policy: No collaboration is permitted.
Grading: This homework will be graded on the scale of 200.

Description: In this homework, you will implement a word learning program for English as a Second Language learners in C programming language using dynamic lists and binary files.

The program's aim is to help an ESL student with synonyms and antonyms. An antonym is of a given word is its opposite in meaning whereas a synonym has the same meaning. For example, "sad" is an antonym of "happy" while "glad" is a synonym.

Your program will do the following:

1. It will read two separate files.
    a. "antonyms.txt" will have at each line a word followed by several antonyms of this word. An example of this file is provided in "antonyms_example.txt" as supplementary material. You are expected to provide a file of your own. This file should have at least 1,000 non-empty lines with an average of at least 2 antonyms per word.
    b. "synonyms.txt" will have at each line a word followed by several synonyms of this word. An example of this file is provided in "synonyms_examples.txt" as supplementary material. You are expected to provide a file of your own. This file should have at least 1,000 non-empty lines with an average of at least 2 synonyms per word.
    c. These files should be read by the program ones as lists.
    d. These lists can be updated by the user through the appropriate interaction. These can be done in two ways:
        i. The program should ask the user if a given answer for the question should be entered as a new entry. For example, the user might have entered an answer that is labeled as incorrect because it is not in the list. In this case, the user optionally can be asked if this should be added as a correct answer.
        ii. The program should have an option for the user to enter a new entry for a word's antonyms or synonyms. This should be a different option than the one above. This will be necessary to add second antonym or synonym (as the first option relies of an incorrect answer to happen).
        iii. In both cases, the updated lists should be reflected in the files.
2. It will at random ask the user a question. Of course, there must be a way to let the user stop the program (add an appropriate dialog for this). If the user wants to continue with the program, there are two types of questions:

    a. What is the synonym of X?
    b. What is the antonym of X?
3. User's answer will be interpreted as follows.
    a. A correct answer should be one of the correct antonyms of synonyms given in the two files. In that case, user's answer should be recorded for the word and its corresponding antonym or synonym (note that there would be more than one for each).
    b. A wrong answer is given when the entered word does not match any of the antonyms of synonyms of the given word. This should be recorded for the user as well.
4. The questions in (2) should not be asked at random but with a bias as the program progresses. So, the selection of the questions should follow the following rules:
    a. In the beginning, any word and any question type can be asked.
    b. As the program progresses (considering after multiple sessions as well – see below for recording multiple sessions and continuing where was left), the words that are not asked (either antonym of synonym) yet should have a higher probability for selection. For example in a 10 word corpus (5 words in the antonym list and 5 in the synonym list), if 2 words are asked 4 times each and the others are not asked at all, the first two words will have ¼ of the chance to be selected compared to the other words. If one word is asked twice for antonyms, then this word will have twice the probability to be selected for a synonym questions compared to the antonym question.
    c. If a word has incorrect answers, it should have higher probability to be picked compared to a one with more correct answers.
5. The program should allow the user to have multiple sessions (at different times). This require the storage of the current state of the user's performance in a file. You need to make sure that the same program in a given directory, might be used by multiple users. This requires:
    a. In the beginning of the program to select from a set of existing users or a new user (in the case of a new user, the user name should be entered).
    b. Each user data will be saved in a binary file named "username.worddat" after the end of each session and loaded back when the same user trains. The size of this file cannot be more than 1.5 times the combined size of the synonym and antonym files.
    c. You can assume that the text and user files are all in the same directory as the executable program.
6. You are expected to use linked lists for your dynamic data needs. The files (antonym, synonym as well as session) should be read only once and should be updated at the end of the program's run. You are not allowed to keep large arrays (on the order of the word counts) neither in stack nor in heap. Smaller fixed size arrays and strings can be kept.

What to hand in: You are expected to hand in a zip or rar file including at least your source code for the main program and any other implementations you have as well as the two word files.

- **HW11_lastname_firstname_studentno.rar / HW11_lastname_firstname_studentno.zip**