BİL 102 – Computer Programming
HW 09

Last Submission Date: 4 June, 2013 at 14:00

In this homework you will implement a poem writer.
Your program will;

- read two sentences from a file,
- create a linked list of words for each sentence
- convert the sentences into a poem with two lines
  - choose one word from each sentence, with the most similar ending (these two words will be the last words of the lines of the poem)
  - arrange the rest of the words of each sentence randomly
- print the poem to a given stream

The main() will take the following input arguments:

- the name of the input file
- the count number of the first sentence
- the count number of the second sentence
- [optional] the name of the out file (if left blank output will be printed to the standard output – pointed by stdout-)

e.g: "./poetWriter in.txt 5 0" should print the poem constructed from the 5th and the 0th sentences of the file "in.txt" in the standard output (i.e. console).

Implement and use at least these functions using the following data structures;

list_node_t
....

index_of_smilar_words_t /*Holds element counts of similar words in linked lists*/
int ind_of_word1 /*element count of the first word in the first list*/ int ind_of_word2 /*element count of the second word in the second list*/

poem_t
list_node_t* line1 list_node_t* line2

poem_t createPoem(const char *fileName, int s1, int s2): Using all required functions below, creates a poem having 2 lines.

int printPoem(const poem_t* poem): Prints the poem and returns the total number of characters in it.

int deletePoem(poem_t* poem): Deallocates all dynamic memory associated with the poem, assigns null pointers to the variables representing lines and returns the number of

words deleted.

char* readSentence(const char *fileName, int s) : Reads the s-th sentence from the file with the given file name and records it to some dynamically allocated memory. (Do not forget to deallocate this memory when you finish your work with it.) Returns null if any problem occurs while reading.

list_node_t* createLL(char *sentence) : Creates and returns a linked list of words from the given sentence. Each word should start with a non-capital letter. **(*)**

int printLine(FILE *out, list_node_t *head_p) : Prints the given line to the out stream. Returns number of characters printed. **(*)**

list_node_t* randomizeSentence(list_node_t *head_p) : Randomly changes the ordering of the words of the sentence and returns a linked list in the new order. Do not use extra space while building the new linkedlist, i.e. use existing nodes. **(You are free to either implement this function or do this operation in createLL function.)**

int checkSimilarity(char *word1, char *word2) : Counts the number of matching characters at the end of two words. **(*)**

index_of_smilar_words_t getSmilarIndexes(list_node_t* head_p1, list_node_t* head_p2): Finds the most similar words in the linked lists and returns their indexes (element counts).

list_node_t* putToEnd(list_node_t *head_p, int index): Puts the indexed element of the linked list to its last position and return the new list. Do not use extra space while building the new linkedlist.

int freeLL(list_node_t *head_p): Deallocates the dynamic memory used in the link list. Returns the number of nodes deleted.**(*)**

(*) You can obtain a bonus of up to **20** points if you implement these functions recursively without changing their prototypes and using helper functions.

General:

1. Obey honor code principles.
2. Obey coding convention.
3. **Avoid code replication** whenever possible.
4. Do not forget to put the required **tags** in the main function.
5. Your submission should include the following file only:
   HW09_<student_name>_<studentSirname>_<student number>.c
6. Deliver the printout of your code **until the last submission date**.
7. Do not use non-English characters in any part of your homework (in body, file name, etc.).