Credit DS-EN-2nd Year & 3rd Year & Supplementary Year / (150 min) | **07.07.2021**

Implement a C-language application that manages students' evaluations for an exam period.

1. Write the source code sequence to create a ***HashTable,*** *referred to it as* ***HT***, that shall store students' evaluations using chaining collision mechanism. The HT structure will store items of type ***Evaluation*.*** The search key used is *examName* **(char*).** Other attributes needed in the Evaluation structure are: *studentName* (**char***), *semester* (valid values 1/2), *finalGrade*(**float**), and 2 other attributes at your choice. The structure will be populated with at least 10 records read from an input file. **(2p)**

   <u>Implementation requirements:</u>
   • Defining ***Evaluation*** structure. (0,25p)
   • Strings from the file must accept blanks when read. (0,25p)
   • No memory leaks. (0,25p)
   • Implementing the logic of creating the HT structure with chaining. (0,75p)
   • Complete and correct implementation, all data inserted into the structure. (0,25p)
   • Testing the implementation in the **main()** function and displaying the content of the newly created HT. (0,25p)

2. Write and call the function for retrieving all the evaluations from the HT structure created above by filtering the elements based on the value of one of the optional attributes whose value is given as a parameter to the function. Evaluations are saved in an array which DOES NOT SHARE heap memory areas with the items from the HT structure. The array is returned in ***main()*** by using the return type or the list of parameters from the function. **(2p)**

   <u>Implementation requirements:</u>
   • Header function definition with I / O parameters, completely and correctly. (0,25p)
   • Items stored in the array do not share heap memory. (0,25p)
   • Implementing the logic of finding and saving the items in the array. (1p)
   • Complete and correct creation of the array. (0,25p)
   • Testing the implementation by calling the function and displaying the results returned after the call. (0,25p)

3. Write and call the function for retrieving the number passed exams for each cluster found in the in the **HashTable**. The result is saved in an array where each item represents the value pair (**cluster index, no. of passed exams**). The array with its size are returned to ***main()*** by using the return type or the formal parameters. The size of the array is dynamically determined. **(2.5p)**

   <u>Implementation requirements:</u>
   • Header function definition with I / O parameters, completely and correctly. (0,25p)
   • Calculating the size of the array. (0,25p)
   • Implementing the logic of identifying the elements of the array. (1,25p)
   • Complete and correct population of the array. (0,25p)
   • Testing the implementation by calling the function and displaying the results returned after the call. (0,50p)

4. Write and call the function for creating **a two-array structure** that stores all the evaluations grouped by semester attribute **(a matrix with two lines, one per each semester)**. The structure is **<u>sharing</u>** memory areas with the HT structure for the evaluations that are stored. The resulted structure is to be returned in ***main()*** and its content displayed at the console. **(2.5p)**

   <u>Implementation requirements:</u>
   • Header function definition with I / O parameters, completely and correctly. (0,25p)
   • Determining the elements that need to be inserted into the matrix. (0,25p)
   • Implementing the logic of creating the new structure; the size of each array in the matrix should be dynamically determined. (1p)
   • Complete and correct population of the matrix **without new memory allocations**. (0,50p)
   • Testing the implementation by calling the function and displaying the results returned after the call. (0,50p)

5. Write and call the functions that free the main structure, **HashTable,** and the arrays involved, as well as all the auxiliary structures used in the implementation of the requirements (if applicable). **(1p)**

   <u>Implementation requirements:</u>
   • Header function definition with I / O parameters, completely and correctly. (0,15p)
   • No memory leaks. (0,15p)
   • Update structure management variables in the main () function. (0,20p)
   • Logical implementation of freeing the data structures. (0,30p)
   • Testing the implementation, complete and correct freeing of structures and displaying the results at the console. (0,20p)
   • Absence of freeing the memory for auxiliary structures used. (-0,20p)

NOTES:

- **Projects with compilation errors won't be evaluated.**
- **Implementations must not contain globally defined or static variables.**
- **Implementations must not use predefined structures such as STL or 3rd party libraries.**
- **Plagiarized implementations will be evaluated with 0 points, regardless of the source.**
- **All requirements must be called and demonstrated in the main () function to be evaluated.**
- **Art. 72 (1) For the following facts, students will be expelled without the right to re-enroll in the Academy of Economic Studies in Bucharest:**
    - **(c) attempting to fraudulently pass examinations or other assessments;**