Name:

Final Exam

Duration: 2 hours 30 minutes

07/30/13

You have to write a program to help the school administrative stuffs to maintain student records. The program will allow the user to do the following tasks from the main control panel:

* 1. Add Student
  2. Search student
  3. Show student list
  4. Find the school topper
  5. Exit

You may use any the class/lab resources but not internet.

**Required Classes**

The following classes need to be created for the exercise. Please note, in the class diagram, + and – denote public and private members respectively. Your class definition should match them accordingly.

**The Student Class**

|  |
| --- |
| Student |
| - m\_id : int  - m\_name : string  - m\_cgpa : double |
| + Student()  +Student(id : int, name : string, cgpa : double)  Add get methods for id, name and cgpa |

**The Sorted List class**

|  |
| --- |
| SortedList |
| - m\_list : Student[]  - m\_size : int  - m\_numOfStudents : int |
| + SortedList()  + SortedLit(size : int)  + getSortedItems() : Student\*  + getNumOfStudents() : int  + addStudents( students : Student[], size : int) : void  + remove(index : int) : void  + searchById(id : int) : Student  - increaseSize() : void  - sort() : void |
| Note:   1. The default constructor will initialize the *m\_list* array with size 5. 2. The *m\_numOfStudents* variable should keep the count of students in the array. 3. *addStudents* function should sort the list after adding the students. 4. *increaseSize* function should increase the list size by 5. 5. *sort* function should sort the student list by their id in ascending order. |

**The Main function**

Create a Main function in main.cpp file and do the following:

1. Load the Students from “Students.txt” file into a *SortedList* object. In the file, each three lines contain the information of one student.
2. Print the main control panel in the console.
3. Take user input and process the request.
4. Keep continuing the steps 2 and 3 unless user presses exit.

**Task precedence**

If you feel that you won’t be able to complete the whole exercise within time, following is the task priority chart. Complete the task with higher priority first:

|  |  |
| --- | --- |
| Priority | Task Name |
| 1 | Add Students (includes sort function) |
| 2 | Search Student |
| 3 | Show all students |
| 4 | Paginate student list |
| 5 | Find the class topper |

**The Sort Function**

You may use any of the sort functions that you learned in the class to sort the student list.

**The Search Function**

Search a student by student id. The search function should be fast, therefore you should avoid linear search. Try for recursive binary search.

*Hint: You may define a recursive function with all the parameters required, and then call the function from SearchById function passing appropriate parameters.*

**Show the student list in paginate format**

While showing student list, if the number of students is more than ten, show first ten at once, then prompt the user to continue or exit. Keep printing the next ten as long as he continues. Sample Output:

Student Name CGPA

101 Adam 3.0

102 Mary 3.1

[List first ten students…]

1. Continue

2. Cancel

**Find the class topper**

Find the student who has the highest CGPA. For this task, you have to devise your own logic. While creating your solution keep the following things in mind:

1. Does it increase the complexity of the program?
2. Does it take extra resources: memory space or looping through certain list?

Note: As your program would run in the computer, memory consumption is less important than the speed of execution. So if your solution requires extra memory however it increases the speed of the program by saving some loop iterations etc., then it’s a better solution.

**Answer the following question**

1. What is the complexity of your program?