Week 9 – SOFT7019 lab session

This week we will utilise an online C IDE called online gdb, please access it at <https://www.onlinegdb.com/>

In the top right corner, you will have the option to select the programming language, please select C.



If you have a problem with this IDE, I would recommend installing CodeBlocks on your local machine. Last week we saw long lag times and errors ub the onlinegdb.com website during some of the lab sessions. I would recommend this guide to follow if installing CodeBlocks:

<https://www.youtube.com/watch?v=GWJqsmitR2I>

# Exercise 1

# **Student database**

1. Define a structure to hold a student’s name (max 10 chars) and the final grade (int).

struct student{

char name[10];

int grade;

};

1. Read the student’s name with scanf.

struct student s;

scanf("%s", s.name);

1. Write a function that sets the student final grade to a certain value

int change\_grade(struct student \*s, int new\_grade){

if (s == NULL) return 1;

s->grade = new\_grade;

return 0;

}

1. Print out the student’s name and final grade.

printf("%s --> %d\n", s.name, s.grade);

Put the above into a complete program.

**Task: student database management**

1. Change the above code so student name is a pointer. The memory should be allocated dynamically with malloc when the student name is read, based on the length of the name

* hint: you should first read the name in a temporary char array, then determine its length.

1. Your student database should be implemented as an array of struct student elements to hold 20 students.
2. Initialise the database, setting name and grade (for as many students as you want, doesn’t have to be 20) using scanf in a for loop.

* remember to allocate memory for the student names

1. Write a function that can add a new entry to the database.

* function should return an error (1) if the database is already full
* the name and grade of the students can be read in the function
* hint: when adding an entry, you modify the array and also its length.

1. Write a function that retrieves an entry from the database, based on its index.

* the function should return a pointer to the entry, so we can then modify the entry in the database.

1. Write a function that prints the entire database.

* each entry should be printed as “index.name –> grade” on a separate line, where index is the index in the database (starts at 0).

1. Write a function that retrieves the student with the lowest grade.

* prototype should be struct student \*get\_lowest(struct student \*students, int num\_students);
* you need to go through the array, keeping track of the student with the lowest grade
* one option is to track the *index* of the lowest grade student as follows:

struct student \*get\_lowest\_index(struct student \*students, int num\_students){

int lowest\_index = 0;

int i;

for (i=1;i<num\_students;i++){

if (students[i].grade < students[lowest\_index].grade){

lowest\_index = i;

}

}

return &students[lowest\_index];

}

* The other option is to use a pointer to track the lowest grade student.

1. Write a function that retrieves the student with the highest grade.
2. Write a function that calculates the average grade.

* prototype should be int get\_average\_grade(struct student \*students, int num\_students);

1. Write a text-based menu interface with the following options:

**1. Print database.**

**2. Add student.**

**3. Change grade.**

**4. Get student with lowest grade.**

**5. Get student with highest grade.**

**6. Get average grade.**

**7. Exit.**

The interface works as follows:

* the menu is displayed for the user
* the user makes a choice by selecting one of the options (use getchar or scanf)
* options 1,2 and 4-6 should call one of the functions defined above
* when option 3 is selected:
  + first the user is asked to input an index from the database (scanf)
  + the function described in sub-task 5 above should be called
  + the selected student will be printed out
  + the user is asked to input the new grade
  + the change\_grade function from the tutorial should be called
* after each operation is complete the menu should be displayed again and all repeats
* the program exits when the user selects option 7.

**Optional extra**

Add a pseudo-histogram feature to the menu. Selecting this will print a horizontal bar chart in the terminalas shown below:

[ 0]:

[10]: #

[20]: ##

[30]:

[40]:

[50]:

[60]: ##

[70]:

[80]: #

[90]:

where the number of#on each line represents the number of students that have grades within that bin (between k and k+9)