**Introduction to Computer Science – 150005**

**Homework Assignment #8**

**Pointers**

**Comments:**

1. Be careful on code readability and appearance (indentation)
2. Make sure to compute exactly what is requested in each question.
3. In any case of incorrect input or another form of error print the message “ERROR” and allow the user to reenter the data. (In this assignment bad inputs are possible in the following cases: bus line number, station number, trip duration (negative numbers) and choosing a number not in the range 0-7 in the main menu.)
4. **Write functions to handle the different tasks the program must do.**
5. Use meaningful variable names
6. Comment each function (including a comment before the function explaining its purpose and how it works.)
7. Submit the solution according to the directions in moodle.

This assignment is meant to check use of pointers. Therefore, it is necessary to use pointers and pointer operators in functions that you write for this assignment. Solutions that use **array subscripting will lose significant points**.

Question 1

Define a gap size of n between 2 integers in an array as the number of elements that appear between the appearance of the first integers and the second. There may be any number of gaps in an array.

For example, in the following array,

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | 2 | 1 | 11 | 2 | 24 | 5 | 2 | 3 | 1 |

the gap between integers 1 and 2 is 0,1,4, or 7.

Gap of 0: (0 intermediary elements)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | 2 | 1 | 11 | 2 | 24 | 5 | 2 | 3 | 1 |

Gap of 1: (1 intermediary elements)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | 2 | 1 | 11 | 2 | 24 | 5 | 2 | 3 | 1 |

Gap of 4: (4 intermediary elements)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | 2 | 1 | 11 | 2 | 24 | 5 | 2 | 3 | 1 |

Gap of 7: (7 intermediary elements)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | 2 | 1 | 11 | 2 | 24 | 5 | 2 | 3 | 1 |

Given an array of integers called **big** and a second array of integers called **small**, **small** is said to be *contained* in **big** if all the integers in **small** are found in **big** in the same order with the same gap size.

Examples:

The array 1,2,8 is contained in the array 11,1,2,8,5 with a gap size of 0.

The array 12,5,4 is contained in the array 14,5,**12**,17,**5**,11,**4**,1,14 with a gap size of 1.

The array 1,2,3 is contained in the array 1,2,2,**1**,5,8,**2**,5,4**,3**,5,8,8,5,1,2,7 with a gap size of 2.

The array 1,2,3 is not contained in the array 1,2,2,3,4,3,3,4

1. Write a function that receives 2 arrays, **big** and **small** and returns the minimum gap size of **small** in **big**. If **small** is not contained in **big**, the function returns -1.
2. Write a main program that inputs the size of **big** (maximum 80) and then the integers to be stored into **big**. Note, X in the second output is the size entered in the first.

Enter the size of big:

Enter X numbers:

Afterward, the main program inputs the size of **small** (maximum 80) and then the integers to be stored into **small**.

Enter the size of small:

Enter X numbers

The main program then calls the function defined in part (a) and prints out the minimum gap size of **small** in **big** after first printing

Size of jump:

As an example:

|  |  |  |
| --- | --- | --- |
| Enter the size of big:  11  Enter 11 numbers:  1,2,2,3,5,8,3,5,1,2,7  Enter the size of small:  3  Enter X numbers:  1,2,3  Size of jump:  -1 | Enter the size of big:  17  Enter 17 numbers:  1,2,2,1,5,8,2,5,4,3,5,8,3,5,1,2,7  Enter the size of small:  3  Enter X numbers:  1,2,3  Size of jump:  2 | Enter the size of big:  5  Enter 5 numbers:  11,1,2,8,5  Enter the size of small:  3  Enter X numbers:  1,2,8  Size of jump:  0 |

**Question 2**

This question should be solved using pointers only!

Some of the classrooms in the Machon have been turned into hybrid classrooms. We would like to manage the registration of these hybrid classes using a database that was created explicitly for this purpose. All of the classrooms are stored in the database in ascending order in which each classroom appears only once. Each classroom has been assigned a five digit (integer) code.

**Managing this database includes:**

* **Checking whether a given classroom is a hybrid** **classroom or not.** (If the classroom appears in the database, then it is indeed a hybrid classroom.)
* Note that the first two digits in the classroom code are the program to which the classroom belongs (10 - Lev Institute, 11 - Lustig, 21 - Tal, 31 – Tvuna, etc ....).

Your program should be able to receive a two digit number as input and **print out all hybrid classrooms that belong to that particular program.**

* **Print out a list of hybrid classrooms** that exist in the Machon.
* **Add a hybrid classroom** to the database.
* **Delete a hybrid classroom** from the database.

The database will be saved in **an array of int** which we will call ***database.*** Each element in the array will hold a classroom number. The maximum number of hybrid classrooms at the Machon is 50 and therefore, the size of the database will be 50.

The database will be sorted in ascending order, according to the classroom codes. Each classroom appears at most once in the database.

The following functions need to be defined:

1. **newClass** - Adding an additional hybrid classroom to the database. The function receives the array (**database**), the number of classrooms that are already stored in the database and the code of the classroom to be added to the database. The function will insert the classroom into the array. The array will remain sorted after the insertion of the new classroom.   
   If the classroom already exists in the array, the array will remain unchanged. There is no need to output any messages to the user.
2. **delClass** - Deletes a classroom from the database. The function receives the array (**database**), the number of classrooms that are stored in the database, and the code of the classroom to be deleted. The function will delete the classroom from the database.  
   If the classroom doesn’t exist in the array, then the array will remain unchanged. There is no need to output any messages to the user.
3. **printAll** - Prints all of the data stored in the database. The function receives the array (database) and the number of classrooms that are already stored in the database. The function prints out all of the classrooms that are stored in the database. Each classroom code printed is separated by a space. If the array is empty, the function will not print anything.
4. **searchClass** - Searches for a classroom within the database. The function receives the array (database), the number of classrooms that are already stored in the database, and the code of the classroom that we are searching for.   
   The function should check if the classroom appears in the database. If the classroom does appear, then the function will return the **address of the array element where the class is found**. If the classroom does not appear in the database, then the function should return NULL.
5. **printCode** - Prints all of the classrooms in the database that belong to a particular program. The function receives the array (database), the number of classrooms that are already stored in the database, and the code of the program that we are searching for. The function prints all of the classrooms in the database that belong to the program.

The main program declares the **database** a.k.a. an array of fifty integers

The main program will include a loop that allows the user to continuously input a choice between 0-5. (This should be handled by an enum.)

The main program will call the appropriate function to handle the user’s request.

If the user enters the number 5, then the program should terminate.

* On input 0: the main program prints the message:

Enter the code of the classroom to add:

Afterwards, it reads in the code of the classroom to add, calls the function **newClass** to add the new classroom to the database, and then calls the function **printAll** to print all of the classrooms that are stored in the database.

* On input 1: the main program print the message:

Enter the code of the classroom to delete:

Afterwards. it reads in the code of the classroom to delete, calls the function **delClass** to delete the classroom from the database, and then calls the function **printAll** to print all of the classrooms that are stored in the database.

* On input 2: the main program prints the message

Enter the code of the classroom to search for:

Afterwards, it reads in the code of the classroom to search for, calls the function **searchClass** to search for the classroom, and prints Found or Not found depending on whether or not the classroom exists or doesn’t exist in the database.

* On input 3: the main program prints the message

Enter the code of the Machon:

Afterwards, it reads in a code (a two digit number) and calls the function **printCode** to print out all the classrooms that are associated with the given program number. If an illegal value is entered, your program should print out an ERROR message and read in a new program code.

* On input 4: the main program calls the function **printAll** which prints out all of the classrooms that are stored in the database.
* On input 5: the main program terminates.

**Additional Notes:**

At the very beginning of execution, the main program should output the following menu:

Enter 0 to add a new classroom.

Enter 1 to delete a hybrid classroom.

Enter 2 to search for a specific classroom.

Enter 3 to print all the classsrooms for a specific Machon.

Enter 4 to print all the hybrid classrooms.

Enter 5 to exit.

Enter your choice:

.

Each time, after the program finishes executing the user’s request, the program should print:

Enter your next choice:

and process the user’s next request. This should continue until the user enters a 5 (to terminate).

Input integrity check:

Unless stated otherwise, after each incorrect input (classroom code that doesn’t contain five digits, program code that doesn’t contain two digits, incorrect menu choice, etc.) the program must print an ERROR message and read in a new value.

Example:

Enter 0 to add a new classroom.

Enter 1 to delete a hybrid classroom.

Enter 2 to search for a specific classroom.

Enter 3 to print all the classsrooms for a specific Machon.

Enter 4 to print all the hybrid classrooms.

Enter 5 to exit.

Enter your choice: 0

Enter the code of the classroom to add: 11111

11111

Enter your next choice: 0

Enter the code of the classroom to add: 44444

11111

44444

Enter your next choice: 0

Enter the code of the classroom to add: 33333

11111

33333

44444

Enter your next choice: 0

Enter the code of the classroom to add: 33555

11111

33333

33555

44444

Enter your next choice: 3

Enter the code of the Machon: 33

33333 33555

Enter your next choice: 4

11111

33333

33555

44444

Enter your next choice: 2

Enter the code of the classroom to search for: 65656

Not found

Enter your next choice: 2

Enter the code of the classroom to search for: 11111

Found

Enter your next choice: 1

Enter the code of the classroom to delete: 33555

11111

33333

44444

Enter your next choice: 0

Enter the code of the classroom to add: 22222

11111

22222

33333

44444

Enter your next choice: 3

Enter the code of the Machon: 5555

ERROR

55

Enter your next choice: 1

Enter the code of the classroom to delete: 33333

11111

22222

44444

Enter your next choice: 1

Enter the code of the classroom to delete: 33333

11111

22222

44444

Enter your next choice: 4

11111

22222

44444

Enter your next choice: 5