Introduction to Computers and Programming LAB-Quiz3 2015/12/30

Time: 2 hrs

※Please create a new folder. Name the folder as: Student ID-Name (XXXXXXX-○○○). Inside the folder, your file format will be Q_1.c, Q_2.c, etc. There will ONLY be a total of 4 .c files in your folder (wrong file name or format will cause score deductions).

※The class is for C language, so do not use C++.

※If any of your program cannot be compiled, you will get zero score for the question.

※Before you use any variables, make sure you have assigned values to them. Some IDE's, such as Dev-C++, may not automatically initialize the variables.

※Your programs will be checked (by a tool) for the programming integrity. Be honest with your own works.

※Your output must comply with the sample output format.

1. SOS from TAs Ver2.0

Implement a toy database system. The required functions are described as what follows.

(1) Add a student and his/her grades to list

Let user input a student's name. Your program should check if the student exists on the list first. If not, ask the user to input the scores for English and Chinese. Calculate the average score with the precision up to the first digit after the decimal point. If the input name has already in the list, your program should warn the user "*Student xxx has already been added.*" Then ask the user to input an instruction again. **If it is a new element, always add it at the correct position in the descending order according to the average**.

(***NOTE for TAs. Program should Check the name first. Modify the output sample too.)

(2) Remove a student from list

Let the user input a student's name that the user would like to remove. Before the user input a student's name, your program should check whether the list is empty. If there is no element in the list, show "*The list has already been empty.*" Then ask user to input an instruction again. **If it is not empty, then remove the element from the list if the input name is in the list; otherwise, show "*No such a student called xxx.*"**

(3) Show the information of the list

If there is not any element in the list, show "The list is empty." Otherwise, **show the first element to the last on the list. Print English score, Chinese score and student Average score (personal). Moreover, print the average grade of all students after the last element (round to the 2rd digit after the decimal point).**

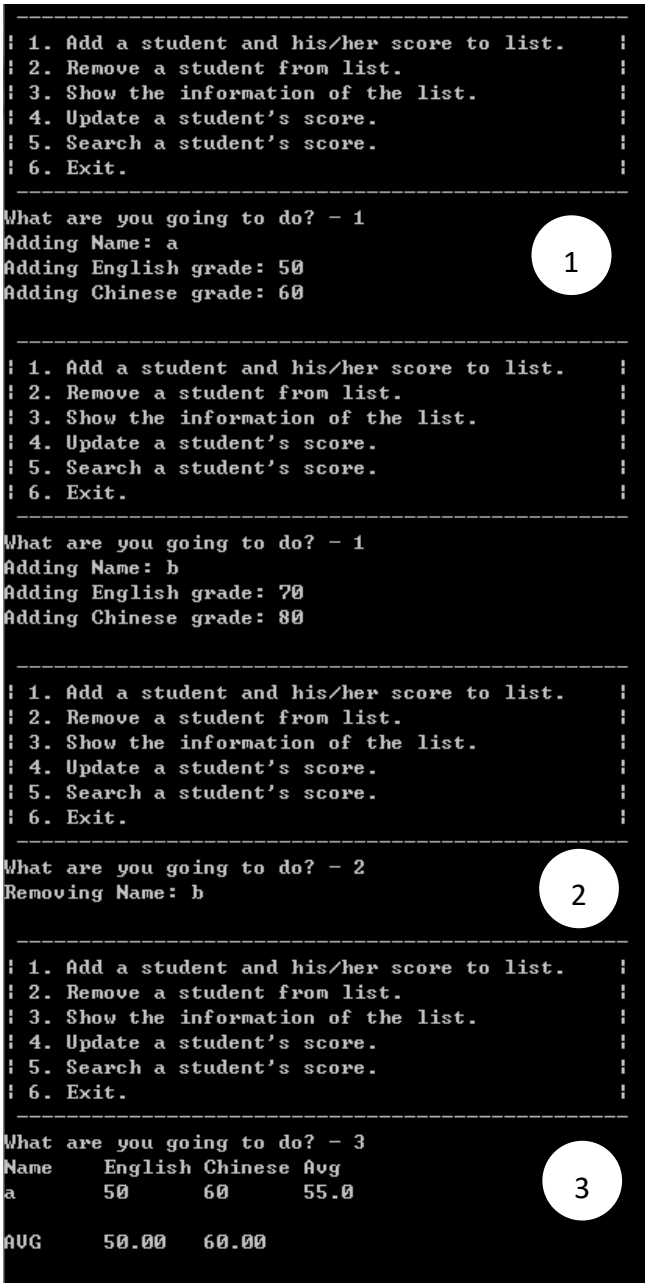
(4) Update a student's score

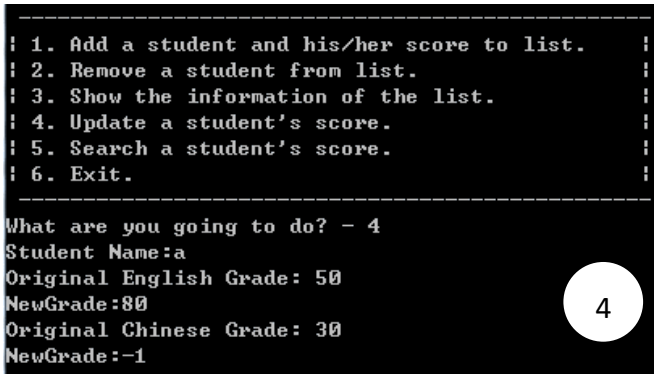
Print the current scores if your program finds the student by the name for update. Then that user can update it. Otherwise, print **"*No such a student called xxx.*". You should ask both English and Chinese score, if the user types "-1", it means skipping over the subject.**

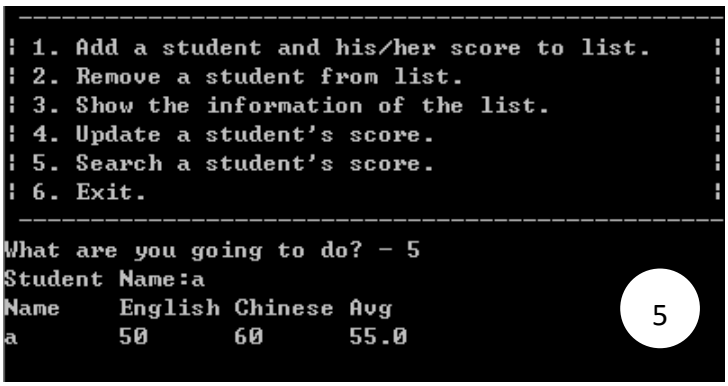
(5) Search a student's score

Print the name and the scores if your program finds the student by the name. Otherwise, print **"*No such a student called xxx.*"**

(6) Exit: Close the program.







2. Dynamic Integer Array Ver2.0

Dynamic memory allocation allows us to allocate a memory block at the execution time, and because the allocated memory block is contiguous, we can use it as an array. However, dealing dynamic memory should be handled carefully. Carelessness can cause memory leak, or even crash a program.

We intend to implement a new data structure called Dynamic Integer Array as well as the related supportive functions for it.

Your program should not crash in any situation!!

Descriptions of Dynamic Integer Array (DynaIntArr):

capacity: the maximum number of elements that we can store

size: the actual number of elements that are stored

data: the pointer to the dynamic allocated memory spaces

Descriptions of supportive functions:

```
DynaIntArr* createArr() {
    use malloc or calloc to allocate a space for DynaIntArr
    set its capacity, size to 0, data to NULL and then return the pointer to it
    return null pointer if allocates memory failed
}
int addAt(struct DynaIntArr *arr, int index , int value) {
    if capacity is not enough
        extend data to size + 1 units of int.
    insert the value to the position of index.
    (valid index range is between 0 to array size, see example below)
      if array is {1, 2, 3}, valid insert indexes are 0, 1, 2, 3
        insert 9 at 0  →  result: {9, 1, 2, 3}
        insert 9 at 1  →  result: {1, 9, 2, 3}
        insert 9 at 2  →  result: {1, 2, 9, 3}
        insert 9 at 3  →  result: {1, 2, 3, 9}
    Increase size by one.
    return 1 if it allocates memory successfully and index is valid; otherwise, return 0.
}
int * getAt(struct DynaIntArr *arr, int index) {
    return "the pointer" to the target value
    (Will return null pointer when the index is invalid)
}
void clearArr(struct DynaIntArr *arr){
    remove all data but do not give back the memory space
    After call clearArr(targetArr),  getOne(targetArr, anyIndex) should always give null pointer.
}
```

int removeAt(struct DynaIntArr *arr, int index) {
    Remove the value of the index from the array
    Move forward all the values behind the index
    return 1 if remove successfully; otherwise, return 0.
    (Assuming that removeOne () is not called frequently, we do not deallocate the memory.)
}
void destroyArr(struct DynaIntArr *arr){
    remove all data and give back the memory space
    after call destroyArr(targetArr), targetArr **will not be accessible** anymore
    that means you have to **free(arr)** and **it's data. (or it will be memory leak)**
}
int reserveArr(DynaIntArr *arr, int size) {
    if the capacity of arr is small than size
        Allocate size units of int.
        Copy the original size ints into the newly allocated spaces.
        return 1 if it **allocates memory successfully**; otherwise, return 0.
    else
        do nothing and return 1
}
int copyArr(DynaIntArr *tar, DynaIntArr *src){
    if the capacity of tar is small than src
        extend the data of tar to the size of src units of int.
    copy all information from src to tar
    return 1 if **copy success**;otherwise, return 0.
}

DynaIntArr* duplicateArr(DynaIntArr *src){
    use malloc or calloc to allocate a space for new DynaIntArr
    copy all information from src to new DynaIntArr
    return 1 if it **allocates memory successfully**;otherwise, return 0.
}

Note:

You have to follow TA's template and finish these nine functions.

You **cannot** declare new **global variable**, **static variable** or **modify exist functions (daLimitCheckMsg, daMemCheckMsg, daIdxCheckMsg, arrModify, printArr and main)** but you can add new functions and local variable

We will use a program to replace main function with another one in order to test the behavior of your functions.

So **DO NOT remove the comment at the end of main, or you will get 0 point.**

Examples: (You can also run “Q2_DynaIntArr2_ans.exe” for more examples.)

```
There are 0 dynamic arrays (100 at most).
 ------------------------------------------------
| 1. Create new Array.                           |
| 2. Modify exist Array.                         |
| 3. Duplicate exist Array.                      |
| 4. Copy one Array into another.                |
| 5. Destroy the Last dynamic array.             |
| 6. Exit.                                       |
 ------------------------------------------------
What do you want to do => 1
```

```
There are 1 dynamic arrays (100 at most).
Array 0 -> size:  0, capacity:  0
```

```
What do you want to do => 2
Please input the index: 0
```

```
size:  0, capacity:  0


 ------------------------------------------------
| 1. Add a value to this dynamic array.          |
| 2. Get the value of this input index.          |
| 3. Remove a value from this dynamic array.     |
| 4. Clear this dynamic array.                   |
| 5. Reserve space.                              |
| 6. Back.                                       |
 ------------------------------------------------
What are you going to do? -> 1
Please input the index: 0
Please input the value: 1
```

```
What are you going to do? -> 1
Please input the index: 1
Please input the value: 2
```

```
What are you going to do? -> 1
Please input the index: 2
Please input the value: 3
```

```
size:  3, capacity:  3
1 2 3
```

```
What are you going to do? -> 1
Please input the index: 2
Please input the value: 9
```

```
size:  4, capacity:  4
1 2 9 3
```

```
What are you going to do? -> 3
Please input the index: 1
```

```
size:  3, capacity:  4
1 9 3
```

```
What are you going to do? -> 5
Please input the size: 2
```

```
size:  3, capacity:  4
1 9 3
```

```
What are you going to do? -> 6
```

```
There are 1 dynamic arrays (100 at most).
Array 0 -> size:  3, capacity:  4
1 9 3
 ---------------------------------------------
| 1. Create new Array.                        |
| 2. Modify exist Array.                      |
| 3. Duplicate exist Array.                   |
| 4. Copy one Array into another.             |
| 5. Destroy the Last dynamic array.          |
| 6. Exit.                                    |
 ---------------------------------------------
What do you want to do => 3
Please input the index: 0
```

```
There are 2 dynamic arrays (100 at most).
Array 0 -> size:  3, capacity:  4
1 9 3
Array 1 -> size:  3, capacity:  3
1 9 3
```

```
What do you want to do => 1
```

```
There are 3 dynamic arrays (100 at most).
Array 0 -> size:  3, capacity:  4
1 9 3
Array 1 -> size:  3, capacity:  3
1 9 3
Array 2 -> size:  0, capacity:  0
```

```
What do you want to do => 4
Please input the target index: 0
Please input the source index: 2
```

```
There are 3 dynamic arrays (100 at most).
Array 0 -> size:  0, capacity:  4

Array 1 -> size:  3, capacity:  3
1 9 3
Array 2 -> size:  0, capacity:  0
```

```
What do you want to do => 5
```

```
There are 2 dynamic arrays (100 at most).
Array 0 -> size:  0, capacity:  4

Array 1 -> size:  3, capacity:  3
1 9 3
```

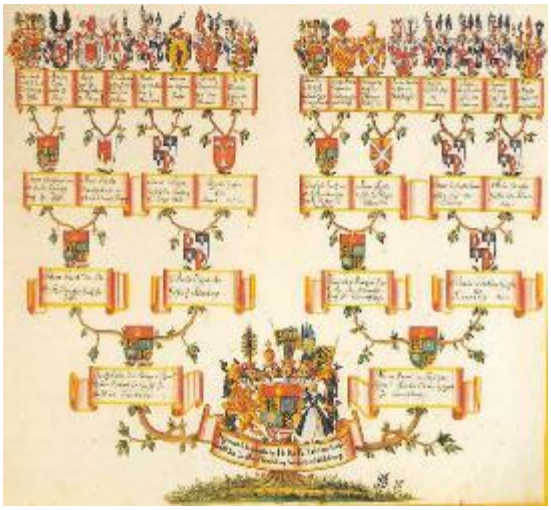
3. Super Calculator

Do you remember the calculator which is able to deal with integer addition, subtraction, multiplication and division in the midterm? Now in this program, you are going to implement **a calculator that can deal with number containing at most 100 digits and 10 operands**. To ease your tasks, there is only one operation in this calculator, addition. The calculator allow user to **input a positive or negative number in each line (means there will be one or more than one operand). If the input number is 0, sum up all the input numbers and output "*The summation is xxx*"**. (xxx is your answer.) Moreover, your program should be able to let users continually enter their inputs.

```
111
0
The summation is 111

111
-111
0
The summation is 0

111
-111
-111
0
The summation is -111

-111
-111
-111
0
The summation is -333

11111111111111
111111111111111111111111111111
111111111111111111111111111111111111111111
111111111111111111111111111111111111
11111111111111111111
111111111111111
99999999999999999999999999999
0
The summation is 11111222222233433334444444444566666666666665
```

4. Family Tree Ver2.0:

A family tree, or pedigree chart, is a chart representing family relationships in a conventional tree structure. The more detailed family trees used in medicine and social work are known as genograms.



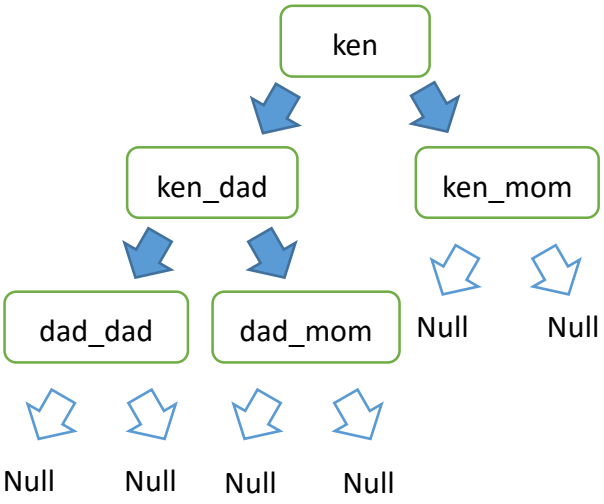
Genealogical data can be represented in several formats, for example as a pedigree or **ancestry chart**. Family trees are often presented with the oldest generations at the top and the newer generations at the bottom. An ancestry chart, which is a tree showing the ancestors of an individual, will more closely resemble a tree in shape, being wider at the top than the bottom. In some ancestry charts, an individual appears on the left and his or her ancestors appear to the right. A descendancy chart, which depicts all the descendants of an individual will be narrowest at the top.

[ref by wiki: https://en.wikipedia.org/wiki/Family_tree]

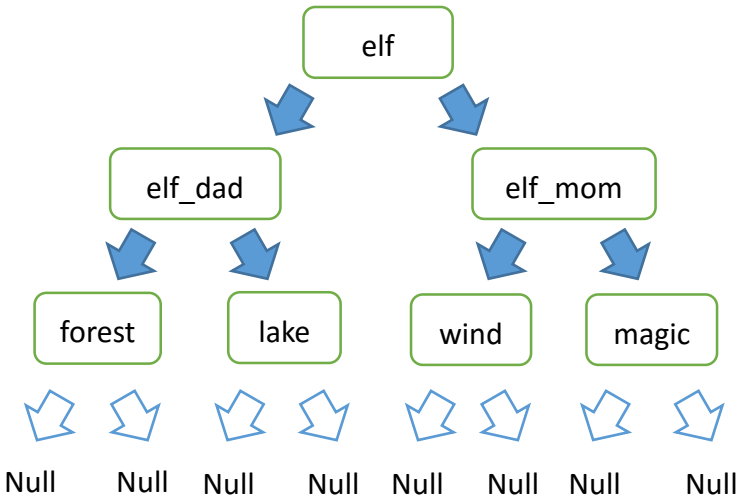
Please use dynamic storage to save family data in family trees. Follow the below steps:

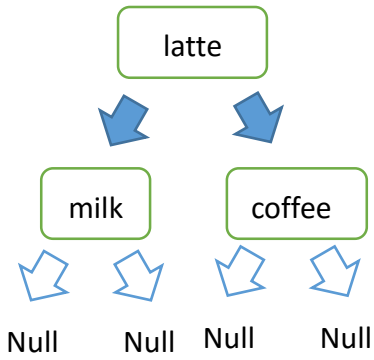
I. Type family data (someone's name & parents' names).
II. If someone's name is matched in one's parent's name in your family trees, add this family data to the corresponding family tree. If someone's name isn't matched in your trees, please create a new tree to save this family data.
III. Keep adding family data until user type "!".
IV. Show one's family tree until user type "!".
V. The number of trees is not greater than 100.
VI. The length of name is not greater than 30.

Note. The test data not include same family data, you don't need handle data collision problem.




```
Please input family data
[name parent1 parent2]
**************************
ken ken_dad ken_mom

Not find ken!! Create new tree 0
**************************
ken_dad dad_dad dad_mom

Add new node to tree 0
**************************
!

show who's ancestors: ken
ken      p1:ken_dad      p2:ken_mom
ken_dad  p1:dad_dad      p2:dad_mom
dad_dad  p1:No Data      p2:No Data
dad_mom  p1:No Data      p2:No Data
ken_mom  p1:No Data      p2:No Data

**************************
show who's ancestors: ken_dad
ken_dad  p1:dad_dad      p2:dad_mom
dad_dad  p1:No Data      p2:No Data
dad_mom  p1:No Data      p2:No Data

**************************
show who's ancestors: ken_mom
ken_mom  p1:No Data      p2:No Data

**************************
show who's ancestors: !

Process returned 0 (0x0)   execution time : 88.052 s
Press any key to continue.
```






```
Please input family data
[name parent1 parent2]
**************************
elf elf_dad elf_mom

Not find elf!! Create new tree 0
**************************
elf_dad forest lake

Add new node to tree 0
**************************
elf_mom wind magic

Add new node to tree 0
**************************
latte milk coffee

Not find latte!! Create new tree 1
**************************
!

show who's ancestors: milk
milk    p1:No Data      p2:No Data
```

```
show who's ancestors: milk
milk    p1:No Data      p2:No Data

**************************
show who's ancestors: latte
latte   p1:milk p2:coffee
milk    p1:No Data      p2:No Data
coffee  p1:No Data      p2:No Data

**************************
show who's ancestors: elf
elf     p1:elf_dad      p2:elf_mom
elf_dad p1:forest       p2:lake
forest  p1:No Data      p2:No Data
lake    p1:No Data      p2:No Data
elf_mom p1:wind p2:magic
wind    p1:No Data      p2:No Data
magic   p1:No Data      p2:No Data

**************************
show who's ancestors: !

Process returned 0 (0x0)   execution time : 168.867 s
Press any key to continue.
```