

Lab 1: Review

1 Pointer

Complete the following given functions

1. Swap 2 given integers.

```
void swap(int* a, int* b)
```

2. Calculate the total value of 2 integers.

```
int* sum(int* a, int* b)
```

3. Input an array with unknown size.

```
void inputArray(int* &a, int &n)
```

4. Print a given array.

```
void printArray(int* a, int n)
```

5. Find the largest value from a given array.

```
int* findMax(int* arr, int n)
```

6. Find the longest ascending subarray from a given array.

```
int* findLongestAscendingSubarray(int* a, int n, int &length)
```

7. Swap 2 given arrays.

```
void swapArrays(int* &a, int* &b, int &na, int &nb)
```

8. Concatenate 2 given arrays.

```
int* concatenate2Arrays(int* a, int* b, int na, int nb, int &nc)
```

9. Given 2 ascending arrays with distinguish elements. Generate a new ascending array with all elements from the given arrays.

```
int* merge2Arrays(int* a, int* b, int na, int nb, int &nc)
```

10. Generate a random matrix with keyboard input size.

```
void generateMatrix1(int** &a, int &length, int &width)
```

11. Given two 1D arrays a and b. Generate the matrix c that $c[i][j] = a[i] * b[j]$.

```
int** generateMatrix2(int* a, int* b, int na, int nb, int &crow, &ccol)
```

12. Swap 2 columns / rows of a given matrix.

```
void swapRows(int** a, int length, int width, int ir1, int ir2)
```

```
void swapColumns(int** a, int length, int width, int ic1, int ic2)
```

13. Generate the transpose matrix of a given matrix.

```
int** transposeMatrix(int** a, int length, int width)
```

14. Concatenate 2 given size-equal matrices, horizontally / vertically.

```
int** concatenate2MatricesH(int** a, int** b,  
                             int length, int width, int &lres, int &wres)
```

```
int** concatenate2MatricesV(int** a, int** b,  
                             int length, int width, int &lres, int &wres)
```

15. Multiple 2 given matrices.

```
bool multiple2Matrices(int**& res, int** a, int **b, int& lengthr, int& widthr, int lengtha,  
                       int widtha, int lengthb, int widthb)
```

16. Given a matrix a. Find the submatrix of a which satisfies keyboard input size and has the largest total value of its elements.

```
int** findSubmatrix(int** a, int length, int width, int &lres, int &wres)
```

From No. 17. to No. 20. are Searching Algorithms. Return the first position found, else, return -1.

17. Sequential Search.

- int LinearSearch(int* a, int n, int key)

18. Sequential Search uses flag.

- int SentinelLinearSearch(int* a, int n, int key)

19. Binary Search in sorted array a.

- int BinarySearch(int* a, int n, int key)

20. Binary Search in sorted array a uses recursion.

- int RecursiveBinarySearch(int* a, int left, int right, int key)

2 Recursion

Complete the following functions using the Recursion technique (*you may declare some sub-functions*):

1. Calculate the sum of $S = 1 + 2 + 3 + \dots + n$.
2. Calculate the factorial $n! = 1 * 2 * 3 * \dots * n$.
3. Calculate x^n .
4. Count the number of digits of a given integer.
5. Verify if every digits of given integer are even.
6. Count the number of common divisor of 2 given integers.
7. Calculate the Greatest common divisor and Least common multiple of 2 given integers.
8. Calculate the reverse value of a given integer.
9. Calculate the i^{th} Fibonacci number.
 - $F_0 = 0, F_1 = 1$
 - $F_n = F_{n-1} + F_{n-2}, (n \geq 2)$
10. * Given 4 single distinguish characters. Print out all possible permutation.
 - Example: ABCD, ABDC, ACBD, ...

3 File Handling

3.1 Data Description

This given data is the anonymized data of the result of the High Graduation Exam 2018 - 2019. The information is provided in the file `"data.txt"`, which has the first few lines as follow:

```

1  Số Báo Danh, Họ và Tên, Toán, Ngữ Văn, Vật Lý, Hóa Học, Sinh Học, Lịch Sử, Địa Lý, GDCD, KHTN, KHXH, Ngoại Ngữ, Ghi Chú, Tỉnh
2  BD1200000,,8.6,6.5,4.0,7.25,5.5,,,,,8.4,N1,BìnhDinh
3  BD1200001,,4.0,5.0,,,4.25,7.0,7.75,,,2.0,N1,BìnhDinh
4  BD1200002,,7.0,6.25,6.0,6.25,6.5,,,,,5.2,N1,BìnhDinh
5  BD1200003,,5.2,5.75,,,5.75,7.25,9.25,,,4.6,N1,BìnhDinh
6  BD1200004,,7.6,6.25,7.0,6.5,4.5,,,,,6.2,N1,BìnhDinh
7  BD1200005,,8.6,6.5,4.0,7.25,5.5,,,,,8.4,N1,BìnhDinh

```

in which:

- The first line provides the included information fields.
- For the next lines, each one is the information of 1 candidate, separated by a comma ",".
- The empty fields mean there is no information. If the empty field is a subject, that equal to a 0.
- The scores in the fields: Natural Sciences (KHTN) and Social Sciences (KHXH) will be instructed in the next part.

3.2 Programming

Given the `Examinee` data structure definition:

```
// Examinee.h
struct Examinee
{
    string id;
    float maths, literature, physics, chemistry, biology, history, geography, civic_education, natural_science,
        social_science, foreign_language;
};
```

Fulfill the following requirements:

1. Read the information of one examinee:

- `Examinee readExaminee(string line_info);`
- **Input:** `line_info` - a line from `"data.txt"` which provides the information of 1 examinee.
- **Output:** Return `Examinee` variable, which stores the info of the given examinee.
- **Note:** The scores of Natural Sciences and Social Sciences column in `data.txt` is not available by default. Calculate the scores for each combination and store them into struct `Examinee`.
 - The score of Natural Sciences combination = physics + chemistry + biology.
 - The score of Social Sciences combination = history + geography + civic education.

2. Read the information of a list of examinees:

- `vector<Examinee> readExamineeList(string file_name);`
- **Input:** `file_name` - path to input file `"data.txt"`.
- **Output:** `vector<Examinee>` variable, which store the info of all examinees from the file.

3. Write scores of examinees to file:

- `void writeScores(vector<Examinee> examinee_list, string out_file_name);`
- **Input:** `examinee_list` - List of examinees.
`out_file_name` - name of file to write.
- **Output:** Write to output file Compulsory Scores (BB), Natural Sciences Scores (KHTN), and Social Sciences (KHXX) of each examinee using the following format:
 - The first line contains header information: `"SDB BB KHTN KHXX"`.
 - Each next line contains info of only one examinee: ID, Compulsory Scores, Natural Sciences Scores, and Social Sciences Scores separated by a single space.
 - The Compulsory Scores = maths + literature + foreign language.
- **Example:**
SDB BB KHTN KHXX
XX001 28.0 29.25 0.0
...
XX999 20.0 0.0 28.75

4 Linkedlist

Given the following Linkedlist definition:

```
struct NODE {  
    int key;  
    NODE* pNext;  
};
```

```
struct List {  
    NODE* pHead;  
    NODE* pTail;  
};
```

Complete the following functions to fulfill the given requirements:

1. Initialize a NODE from a given integer:
 - `NODE* createNode(int data)`
2. Initialize a List from a give NODE:
 - `List* createList(NODE* pNode)`
3. Insert an integer to the head of a given List:
 - `bool addHead(List* &L, int data)`
4. Insert an integer to the tail of a given List:
 - `bool addTail(List* &L, int data)`
5. Remove the first NODE of a given List:
 - `void removeHead(List* &L)`
6. Remove the last NODE of a given List:
 - `void removeTail(List* &L)`
7. Remove all NODE from a given List:
 - `void removeAll(List* &L)`
8. Remove node before the node has val value in a given List:
 - `void removeBefore(List* L, int val)`
9. Remove node after the node has val value in a given List:
 - `void removeAfter(List* L, int val)`
10. Insert an integer at a position of a given List:
 - `bool addPos(List* &L, int data, int pos)`
11. Remove an integer at a position of a given List:
 - `void removePos(List* &L, int pos)`
12. Insert an integer before a value of a given List:
 - `bool addBefore(List* L, int data, int val)`
13. Insert an integer after a value of a given List:
 - `bool addAfter(List* L, int data, int val)`
14. Print all elements of a given List:
 - `void printList(List* L)`
15. Count the number of elements List:
 - `int countElements(List* L)`
16. Create a new List by reverse a given List:
 - `List* reverseList(List* L)`
17. Remove all duplicates from a given List:
 - `void removeDuplicate(List* &L)`
18. Remove all key value from a given List:
 - `bool removeElement(List* &L, int key)`

5 Doubly Linkedlist

Following is representation of a doubly linked list:

```
struct DNODE {
    int key;
    DNODE* pNext;
    DNODE* pPrev;
};
```

```
struct DList {
    DNODE* pHead;
    DNODE* pTail;
};
```

Implement functions to execute the operations from singly linkedlist section 4.

6 Stack - Queue

Following is the representation of a Singly linked list node:

```
struct NODE {
    int key;
    NODE* pNext;
};
```

Utilize the Linked list above, define the data structure of Stack and Queue, then implement functions to execute the following operations:

1. Stack

- **Initialize** a stack from a given key.
- **Push** a key into a given stack.
- **Pop** an element out of a given stack, return the key's value.
- **Count** the number of elements of a given stack.
- Determine if a given stack **is empty**.

2. Queue

- **Initialize** a queue from a given key.
- **Enqueue** a key into a given queue.
- **Dequeue** an element out of a given queue, return the key's value.
- **Count** the number of element of a given queue.
- Determine if a given queue **is empty**.