CS 201, Spring 2024

Homework Assignment 3

Due: 23:59, April 28, 2024

1 Introduction

In this homework you will implement a system to store biological pathways in an organism using linked lists. Genes in an organism encode proteins. Multiple genes can encode subunits of a single protein. Proteins, which as a group performing a certain task, forms biological pathways. The system you will implement consists of pathways with unique IDs, proteins with unique IDs, and genes with unique IDs throughout the system. Note that these ID types are independent of each other. For instance, there cannot be more than one pathway with ID 6, but there can be a gene with ID 6 and a protein with ID 6. They would be different entities.

In this system, you will store a linked list of pathways. For each pathway, you will store a linked list of proteins participating in that pathway. For each protein, you will store a linked list of genes encoding that protein. In your implementation, you MUST use **sorted linked lists** for storing the pathways, proteins and genes.

1.1 Pathways

Each pathway has a unique ID and a name. You need to use a sorted linked list to store the items with respect to ascending pathway IDs. The required functions are as follows:

- `addPathway`: Adds a new pathway to the system with an ID and a name. Since IDs must be unique, the system must check whether or not the specified pathway ID already exists, and if it exists, it must not allow the operation and display a warning message. Initially, a pathway does not have any proteins. Example log messages:
  - Added pathway 6.
  - Cannot add pathway. There is already a pathway with ID 6.
- `removePathway`: The system will allow removing pathways. If you remove the pathway, all related proteins and genes are removed as well. If the pathway does not exist, the system must display a warning message. Example log messages:
  - Removed pathway 6.
  - Cannot remove pathway. There is no pathway with ID 6.
- `printPathways`: Prints IDs, pathway names, and also the total number of proteins and encoder genes that each pathway contains in the system. If there is no pathway in the

system, give a warning message. The entries must be in ASCENDING ORDER with respect to the IDs. Example log messages:

  - There are no pathways to show.
  - Pathways in the system:
    Pathway 6 : Apoptosis Signaling Pathway (5 Proteins) (24 Genes)
    Pathway 13 : Notch Signaling Pathway (2 Proteins) (9 Genes)
    Pathway 19 : BLKNT Regulatory Pathway (9 Proteins) (48 Genes)

1.2 Proteins

Each protein is identified by a unique ID. It also has a list of encoder gene IDs. You need to use a sorted linked list to store the items with respect to ascending protein IDs. Each protein can participate in a single pathway. The required functions are as follows:

- `addProtein`: Adds a new protein to the pathway with an ID. The system must first check whether pathway ID exists. If it exists, then it should check whether specified protein ID exists in any pathway. If there is already a protein with same ID in a pathway or the specified pathway does not exist, it must not allow the operation and display a warning message. Initially, proteins does not contain any genes. Example log messages:
  - Added protein 6 to pathway 4.
  - Cannot add protein. There is a pathway having a protein with ID 6.
  - Cannot add protein. There is no pathway with ID 4.
- `removeProtein`: The system will allow removing proteins from pathways. If the pathway does not exist or the pathway does not contain the specified protein, the system must display a warning message. Example log messages:
  - Removed protein 6 from pathway 4.
  - Cannot remove protein. Pathway 4 does not have a protein with ID 6.
  - Cannot remove protein. There is no pathway with ID 4.
- `printProteinsInPathway`: Prints the list of proteins participating in the pathway. The message should include protein IDs and responsible gene IDs. If the specified pathway does not exist or if there is no protein in a pathway, give a warning message. The entries should be in ASCENDING ORDER according to IDs. Example log messages:
  - Cannot print proteins. There is no pathway with ID 4.
  - There are no proteins to show in pathway 4.
  - Proteins in pathway 4:
    Protein ID : 5, Gene IDs : [6, 7, 8, 9]
    Protein ID : 14, Gene IDs : [2, 5, 12]
    Protein ID : 24, Gene IDs : None

1.3 Genes

Proteins carry the IDs and names of genes which encode them. Each gene can encode a subunit of a single protein. In other words, there should not be genes with same IDs that encode different proteins.

- `addGene`: Adds a new gene to the protein. If the gene was already added for any protein, or if the specified protein ID does not exist, give a warning message and do not allow the operation. Example log messages:
  - Added gene 23 to protein 6.
  - Cannot add gene. Gene 23 is already in use.
  - Cannot add gene. There is no protein with ID 6.
- `removeGene`: Removes the gene from the specified protein. If the protein does not exist or the protein does not contain the specified gene, give a warning message. Example log messages:
  - Removed gene 23 from protein 6.
  - Cannot remove gene. There is no gene 23 encoding protein 6.
  - Cannot remove gene. There is no protein with ID 6.
- `printGenesOfProtein`: Prints the names and IDs of the encoding genes for the given protein. If the specified protein does not exist, give a warning message. The entries should be in ASCENDING ORDER according to IDs. Example log messages:
  - Cannot print genes. There is no protein with ID 4.
  - There are no genes to show in protein 4.
  - Genes in protein 4:
    Gene 23 : BLKNT
    Gene 46 : NLFTR
    Gene 89 : RPLA

Below is the required public part of the `BiologicalPathway` class that you must write in this assignment. The name of the class must be `BiologicalPathway`, and must include these public member functions. The interface for the class must be written in the file called `BiologicalPathway.h` and its implementation must be written in the file called `BiologicalPathway.cpp`. You can define additional public and private member functions and data members in this class. You can also define additional classes in your solution and implement them in separate files.

```
class BiologicalPathway {
public:
    BiologicalPathway();
    ~BiologicalPathway();

    void addPathway( const int pathwayId, const string pathwayName );
    void removePathway( const int pathwayId );
    void printPathways() const;

    void addProtein( const int proteinId, const int pathwayId );
    void removeProtein( const int proteinId, const int pathwayId );
    void printProteinsInPathway( const int pathwayId ) const;

    void addGene( const int geneID, const string geneName, const int proteinId );
    void removeGene( const int geneID, const int proteinId );
    void printGenesOfProtein( const int proteinId ) const;

};
```

Here is an example test program that uses this class and the corresponding output. We will use a similar program to test your solution so make sure that the name of the class is `BiologicalPathway`, its interface is in the file called `BiologicalPathway.h`, and the required functions are defined as shown above. Your implementation **MUST** use exactly the same format given in the example output to display the messages expected as the result of the defined functions.

Example test code:

```
int main() {

    BiologicalPathway biologicalPathway;

    biologicalPathway.addPathway(915, "Apoptosis Signaling Pathway");
    biologicalPathway.addPathway(915, "Apoptosis Signaling Pathway");
    biologicalPathway.addPathway(908, "BLKNT Regulatory Pathway");
    biologicalPathway.addPathway(901, "Notch Signaling Pathway");
    cout << endl;

    biologicalPathway.addProtein(56,915);
    biologicalPathway.addProtein(51,915);
    biologicalPathway.addProtein(59,915);
    biologicalPathway.addProtein(59,999);
    cout << endl;

    biologicalPathway.addProtein(63,908);
    biologicalPathway.addProtein(67,908);
```

```
    biologicalPathway.addProtein(56,908);
    cout << endl;

    biologicalPathway.addGene(41,"TYUI",63);
    biologicalPathway.addGene(48,"BNMF",63);
    biologicalPathway.addGene(13,"CFTR",51);
    biologicalPathway.addGene(13,"TYOT",56);
    biologicalPathway.addGene(11,"TYQT",98);
    biologicalPathway.addGene(16,"RTYU",51);
    biologicalPathway.addGene(18,"REYU",59);
    biologicalPathway.addGene(21,"TYUJ",59);
    biologicalPathway.addGene(13,"YUIN",67);
    biologicalPathway.addGene(29,"BNMB",59);
    cout << endl;

    biologicalPathway.printProteinsInPathway(1005);
    biologicalPathway.printProteinsInPathway(915);
    biologicalPathway.printProteinsInPathway(901);
    cout << endl;

    biologicalPathway.printGenesOfProtein(59);
    biologicalPathway.printGenesOfProtein(56);
    biologicalPathway.printGenesOfProtein(11);
    cout << endl;

    biologicalPathway.removeGene(21,89);
    biologicalPathway.removeGene(21,59);
    biologicalPathway.removeGene(13,51);
    biologicalPathway.removeGene(895,51);
    cout << endl;

    biologicalPathway.removeProtein(63,908);
    biologicalPathway.removeProtein(63,908);
    biologicalPathway.removeProtein(56,1005);
    cout << endl;

    biologicalPathway.printPathways();
    cout << endl;

    biologicalPathway.removePathway(915);
    biologicalPathway.removePathway(915);
    cout << endl;

    biologicalPathway.printPathways();
    cout << endl;

```

```
    biologicalPathway.removePathway(901);
    biologicalPathway.removePathway(908);
    cout << endl;

    biologicalPathway.printPathways();

    return 0;
}
```

Output of the example test code:

```
Added pathway 915.
Cannot add pathway. There is already a pathway with ID 915.
Added pathway 908.
Added pathway 901.

Added protein 56 to pathway 915.
Added protein 51 to pathway 915.
Added protein 59 to pathway 915.
Cannot add protein. There is no pathway with ID 999.

Added protein 63 to pathway 908.
Added protein 67 to pathway 908.
Cannot add protein. There is a pathway having a protein with ID 56.

Added gene 41 to protein 63.
Added gene 48 to protein 63.
Added gene 13 to protein 51.
Cannot add gene. Gene 13 is already in use.
Cannot add gene. There is no protein with ID 98.
Added gene 16 to protein 51.
Added gene 18 to protein 59.
Added gene 21 to protein 59.
Cannot add gene. Gene 13 is already in use.
Added gene 29 to protein 59.

Cannot print proteins. There is no pathway with ID 1005.
Proteins in pathway 915:
Protein ID : 51, Gene IDs : [13, 16]
Protein ID : 56, Gene IDs : None
Protein ID : 59, Gene IDs : [18, 21, 29]
There are no proteins to show in pathway 901.

Genes in protein 59:
GENE 18 : REYU
GENE 21 : TYUJ
```

```
GENE 29 : BNMB
There are no genes to show in protein 56.
Cannot print genes. There is no protein with ID 11.

Cannot remove gene. There is no protein with ID 89.
Removed gene 21 from protein 59.
Removed gene 13 from protein 51.
Cannot remove gene. There is no gene 895 encoding protein 51.

Removed protein 63 from pathway 908.
Cannot remove protein. Pathway 908 does not have a protein with ID 63.
Cannot remove protein. There is no pathway with ID 1005.

Pathways in the system:
Pathway 901 : Notch Signaling Pathway (0 Proteins) (0 Genes)
Pathway 908 : BLKNT Regulatory Pathway (1 Proteins) (0 Genes)
Pathway 915 : Apoptosis Signaling Pathway (3 Proteins) (3 Genes)

Removed pathway 915.
Cannot remove pathway. There is no pathway with ID 915.

Pathways in the system:
Pathway 901 : Notch Signaling Pathway (0 Proteins) (0 Genes)
Pathway 908 : BLKNT Regulatory Pathway (1 Proteins) (0 Genes)

Removed pathway 901.
Removed pathway 908.

There are no pathways to show.
```

2 Specifications

1. You ARE NOT ALLOWED to modify the given parts of the header file. You MUST use **sorted linked lists** in your implementation. You will get no points if you use dynamic or fixed-sized arrays or any other data structures such as vectors/arrays/lists from the standard library. However, if necessary, you may define additional data members and member functions. You may also define additional classes.

2. Moreover, you ARE NOT ALLOWED to use any global variables or any global functions.

3. Output message for each operation MUST match the format shown in the output of the example code.

4. Your code MUST NOT have any memory leaks. You will lose points if you have memory leaks in your program even though the outputs of the operations are correct. To detect

memory leaks, you may want to use Valgrind which is available at http://valgrind.org.

3 Submission

1. In this assignment, you must have separate interface and implementation files (i.e., separate `.h` and `.cpp` files) for your class. Your class name MUST BE `BiologicalPathway` and your file names MUST BE `BiologicalPathway.h` and `BiologicalPathway.cpp`. Note that you may write additional class(es) in your solution.

2. The code (`main` function) given above is just an example. We will test your implementation using different scenarios, which will contain different function calls. Thus, do not test your implementation only by using this example code. We recommend you to write your own driver files to make extra tests. However, you MUST NOT submit these test codes (we will use our own test code). In other words, do not submit a file that contains a function called `main`.

3. You should put all of your `.h` and `.cpp` files into a folder and zip the folder (in this zip file, there should not be any file containing a `main` function). The name of this zip file should conform to the following name convention: secX-Firstname-Lastname-StudentID.zip where X is your section number. The submissions that do not obey these rules will not be graded.

4. Make sure that each file that you submit (each and every file in the archive) contains your name, section, and student number at the top as comments.

5. You are free to write your programs in any environment (you may use Linux, Windows, MacOS, etc.). On the other hand, we will test your programs on "dijkstra.ug.bcc.bilkent.edu.tr" and we will expect your programs to compile and run on the dijkstra machine. If we could not get your program properly work on the dijkstra machine, you would lose a considerable amount of points. Thus, we recommend you to make sure that your program compiles and properly works on dijkstra.ug.bcc.bilkent.edu.tr before submitting your assignment.

6. This assignment is due by 23:59 on Sunday, April 28, 2024. You should upload your work to Moodle before the deadline. No hardcopy submission is needed. The standard rules about late homework submissions apply. Please see the course home page for further discussion of the late homework policy.

7. We use an automated tool as well as manual inspection to check your submissions against plagiarism. Please see the course home page for further discussion of academic integrity and the honor code for programming courses in our department.

8. This homework will be graded by your TA **Mehmet Alper Yilmaz** (mehmet.yilmaz@bilkent.edu.tr). Thus, you may ask your homework related questions directly to him. There will also be a forum on Moodle for questions.