Sabancı University

Faculty of Engineering and Natural Sciences

CS204 Advanced Programming

Fall 2020-2021

Homework #2

Due: 01/11/2020 - 23:55

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!

You HAVE TO write down the code on your own.
You CANNOT HELP any friend while coding.
Plagiarism will not be tolerated!!

1 Introduction

The aim of this homework is to use linked lists and doubly linked lists. You will employ these structures to implement a metro line/stop management system. Since the number of metro stops cannot be known, you will use dynamic memory allocation, i.e., new/delete operations to create and delete these structures. Don't forget, we don't want memory leaks in the software so each memory region allocated by a new operation should be freed by a delete operation once it is not necessary anymore.

2 Sample Input File

They want you to implement a management system. Before, the people of the metro line company were keeping the track of their data in text files. A sample file is given below:

```
M1A,Kirazli,Otogar,Topkapi,Aksaray,Yenikapi
M1B,Ataturk_Havalimani,Atakoy,Bahcelievler,Zeytinburnu,Otogar
MAR,Kazlicesme,Yenikapi,Beyazit,Sirkeci,Uskudar,Marmaray
M2,Yenikapi,Beyazit,Tunel,Taksim,Mecidiyekoy,Levent,ITU,Haciosman
T1,Bagcilar,Akincilar,Gungoren,Zeytinburnu,Topkapi,Findikzade,Sirkeci,Karakoy,Kabatas
M4,Kadikoy,Marmaray,Uzuncayir,Kozyatagi,Maltepe,Huzurevi,Kartal,Pendik
M6,Levent,Nispetiye,Etiler,Bogazici
34G,Beylikduzu,Guzelyurt,Florya,Atakoy,Bahcelievler,Zeytinburnu,Topkapi,Edirnekapi,
Darulaceze,Mecidiyekoy,Sehitler_Koprusu,Uzuncayir
F1,Taksim,Kabatas
F2,Tunel,Karakoy
T2,Tunel,Taksim
M3,Basaksehir,Ikitelli,Mahmutbey,Bagcilar
```

Each metro line is given at a line in the file and commas separate all the metro line items. The first item is the name of the metro line, and the rest are the names of the metro stops except the last item. The last item is to denote that the metro line information ends (i.e., the line ends just after it). You don't need to check the consistency of the information in the input file. Each line will always contain the metro line name, at least one metro stop, and the termination character. In addition, metro line and metro stop names will always be a single word. Don't worry about these.

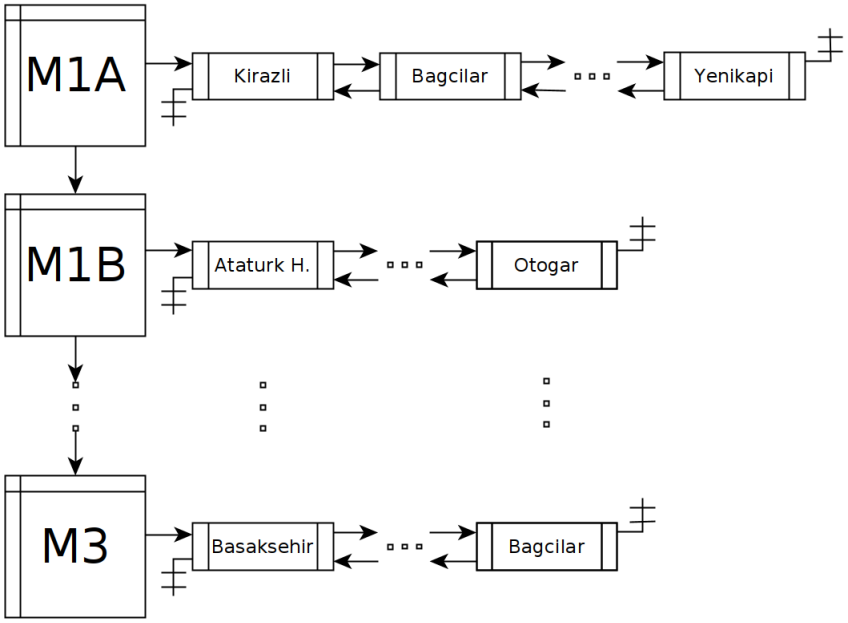
They were also keeping the costs of traveling between stations in a separate file. A sample file is given below:

```
M1A,8,11,6,9
M1B,7,7,6,5
MAR,4,4,20,3,3
M2,7,6,7,6,7,6
T1,3,3,2,1,3,4,2,3
M4,8,6,7,8,9,11,2
M6,8,7,8
34G,5,4,4,3,6,4,5,7,21,4,5
F1,3
F2,2
T2,7
M3,8,6,8
```

The cost of traveling between stations are given at each line. Each value represents a transaction between two stops, therefore there are *#of stations - 1* values at each line.

3 The Data Structure

You will use a 2D metro line table. The lines will be stored in a single linked list and the stops of a metro line will be stored in a double linked list. Here is a visual description of the table.



For metro lines, having same stop means there is a transfer between these lines. Therefore one can continue with other line when s/he arrives such stop. For example in the figure, one can go to Yenikapi from Basaksehir by transferring at M3 Bagcilar to M1A Bagcilar.

The only thing you need to be careful about is an empty metro line. They cannot exist. Each metro line needs to have at least one stop. So when the last metro stop of a metro line is deleted (see the rest of the assignment), the metro line also needs to be deleted.

4 The Main Menu

The node/list structures and functions to print the main menu and to process the input are given below to make your job easier. You need to use them as is but you may want to (we prefer that you do) to add appropriate constructors and members to the struct definitions. The `consistencyCheck()` function is checking if your doubly linked lists are always correct.

```
bool consistencyCheck() {
  metroLine* currBL = head;
  while(currBL) {
    metroStop* currBS = currBL->metroStops;
    while(currBS) {
      metroStop* rightBS = currBS->right;
      if(rightBS && rightBS->left != currBS) {
  cout << "Inconsistency for "
       << currBL->metroNumber
       << " "
       << currBS->metroStopName << endl;
  return false;
      }
      currBS = currBS->right;
    }
    currBL = currBL->next;
  }
  return true;
}
```

```
struct metroStop {
  string metroStopName;
  metroStop *left;
  metroStop *right;
};

struct metroLine {
  string metroNumber;
  metroLine *next;
  metroStop *metroStops;

  vector<double>costs;
};
```

```
void processMainMenu() {
  char input;
  do{
    if(!consistencyCheck()) {
      cout << "There are inconsistencies. Exit." << endl;
      return;
    }
    printMainMenu();
    cout << "Please enter your option " << endl;
    cin >> input;
    switch (input) {
    case '0':
      cout << "Thanks for using our software" << endl;
      return;
    case '1':
      printmetroLinesDetailed();
      break;
    case '2':
      addmetroStop();
      break;
    case '3':
      deletemetroStop();
      break;
    case '4':
      pathfinder();
      break;
    default:
      cout << "Invalid option: please enter again" << endl;
    }
  } while(true);
}
```

```
void printMainMenu() {
  cout << endl;
  cout <<"I*********************************************I"<<endl
       <<"I               0 - EXIT PROGRAM              I"<<endl
       <<"I               1 - PRINT LINES               I"<<endl
       <<"I               2 - ADD METRO STOP            I"<<endl
       <<"I               3 - DELETE METRO STOP         I"<<endl
       <<"I               4 - PATH FINDER               I"<<endl
       <<"I*********************************************I"<<endl
       <<">>";
  cout << endl;
}
```

Here we describe the main menu functions with some examples for a better understanding:

4.1 Print Lines

This menu item will print the metro line table. Each metro line will be printed in a line. The metro line name will be the first item in the line followed by a ':'. Then " $<-$" and "$->$ " symbols will be used to separate the metro stop names with cost associated with the route between them will be used to separate the metro stop names.

Execution:

```
Please enter your option
1
M1A: Kirazli<-8->Otogar<-11->Topkapi<-6->Aksaray<-9->Yenikapi
M1B: Ataturk_Havalimani<-7->Atakoy<-7->Bahcelievler<-6->Zeytinburnu<-5->Otogar
MAR: Kazlicesme<-4->Yenikapi<-4->Beyazit<-20->Sirkeci<-3->Uskudar<-3->Marmaray
M2:  Yenikapi<-7->Beyazit<-6->Tunel<-7->Taksim<-6->Mecidiyekoy<-7->Levent<-6->ITU<-7->Haciosman
T1:  Bagcilar<-3->Akincilar<-3->Gungoren<-2->Zeytinburnu<-1->Topkapi <-3->Findikzade<-4->Sirkeci<-2->Karakoy
<-3->Kabatas
M4:  Kadikoy<-8->Marmaray<-6->Uzuncayir<-7->Kozyatagi<-8->Maltepe<-9-> Huzurevi<-11->Kartal<-2->Pendik
M6:  Levent<-8->Nispetiye<-7->Etiler<-8->Bogazici
34G: Beylikduzu<-5->Guzelyurt<-4->Florya<-4->Atakoy<-3->Bahcelievler<-6->Zeytinburnu<-4->Topkapi<-5->Edirnekapi
<-7->Darulaceze<-21->Mecidiyekoy<-4->Sehitler_Koprusu<-5->Uzuncayir
F1:  Taksim<-3->Kabatas
F2:  Tunel<-2->Karakoy
T2:  Tunel<-7->Taksim
M3:  Basaksehir<-8->Ikitelli<-6->Mahmutbey<-8->Bagcilar
```

4.2 Add Metro Stop

This menu item will be used to add a new metro stop. First an existing metro line name will be asked. If it does not exist in the table then main menu will appear (Execution 1). If it exists, the metro line's information will be shown (Execution 2&3) and the new metro stop name will be requested. If the metro stop already exists in the metro line main menu will appear (Execution 2). -You should actually call the main menu and ask for input. This is not given here to save space - Otherwise, the previous metro stop name for the new metro stop will be asked. 0 is reserved to add the new metro stop to the doubly linked list as the first one (Execution 3). If the name does not exist then it will be asked again (Execution 3). When an existing metro stop is given, the program inserts the new metro stop just after it and show the updated metro line table. While adding new metro lines, costs between the stops should also be taken into consideration. After giving valid line and stop names, program should ask for *cost1* and *cost2* that will emerge between previous and next stops of the newly added stop (Execution3). If new stop is at the beginning of the line, program should use *cost1* as cost between new first stop and the old one (Execution 4). If the new stop is at the end of the line, program again should use *cost1* as cost between new last stop and the old one (Execution 5). After adding stop, program should print out new state of the line.

Execution 1

```
Please enter your option
2
Enter the name of the metro line to insert a new metro stop (0 for main menu)
```

```
myMetroLine2
Metro line cannot be found.  Going back to previous menu.
```

Execution 2

```
Please enter your option
2
Enter the name of the metro line to insert a new metro stop (0 for main menu)
myMetroLine
The metro line information is shown below
myMetroLine:  metroStop1<-3->metroStop2<-5->metroStop3
Enter the name of the new metro stop
metroStop1 Metro stop already exists.  Going back to previous
menu.
```

Execution 3

```
Please enter your option
2
Enter the name of the metro line to insert a new metro stop (0 for main menu)
M1B
The metro line information is shown below
M1B: Ataturk_Havalimani<-7->Atakoy<-7->Bahcelievler<-6->Zeytinburnu<-5->Otogar
Enter the name of the new metro stop
Bakirkoy
Enter the name of the previous metro stop for the new one (0 to put the new one as the first metro stop)
Bahcekoy
Enter new cost 1
3
Enter new cost 2
4
Metro stop does not exist.  Typo?  Enter again (0 for main menu)
Bahcelievler
M1B: Ataturk_Havalimani<-7->Atakoy<-7->Bahcelievler<-3->Bakirkoy<-4->Zeytinburnu<-5->Otogar
```

Execution 4

```
Please enter your option
2
Enter the name of the metro line to insert a new metro stop (0 for main menu)
M1B
The metro line information is shown below
M1B: Ataturk_Havalimani<-7->Atakoy<-7->Bahcelievler<-6->Zeytinburnu<-5->Otogar
Enter the name of the new metro stop
Yesilkoy
Enter the name of the previous metro stop for the new one (0 to put the new one as the first metro stop)
0
Enter new cost 1
2
Enter new cost 2
9
M1B: Yesilkoy<-2->Ataturk_Havalimani<-7->Atakoy<-7->Bahcelievler<-6->Zeytinburnu<-5->Otogar
```

Execution 5

```
Please enter your option
2
Enter the name of the metro line to insert a new metro stop (0 for main menu)
M1B
The metro line information is shown below
M1B: Ataturk_Havalimani<-7->Atakoy<-7->Bahcelievler<-6->Zeytinburnu<-5->Otogar
Enter the name of the new metro stop
```

```
Sisli
Enter the name of the previous metro stop for the new one (0 to put the new one as the first metro stop)
Otogar
Enter new cost 1
14
Enter new cost 2
3
M1B: Ataturk_Havalimani<-7->Atakoy<-7->Bahcelievler<-6->Zeytinburnu<-5->Otogar<-14->Sisli
```

4.3 Delete Metro Stop

This menu item will be used to delete an existing metro stop. First a metro line will be requested as an input. If the line does not exist, main menu will appear (Execution 1). Otherwise, the metro line information will be shown (Execution 2) and the metro stop name will be requested (0 is reserved to go back to the main menu). If the metro stop does not exist in the metro line, the name will be asked again (Execution 2). If the metro stop exists, it will be deleted and the updated metro line will be shown before the appearance of the main menu (Execution 2). Because of the deleted stop, costs between stops should be rearranged. In contrast to add stop, when a stop is deleted, the transaction between it's previous stop and next stop should have their costs merged (Execution 2). If the deleted stop is at the beginning or at the end of it's line, the cost of the paths that leading to them should be removed.(Execution 3)

Execution 1

```
Please enter your option
3
Enter the name of the metro line to delete a new metro stop (0 for main menu)
myMetroLine2
Metro line cannot be found.  Going back to previous menu.
```

Execution 2

```
Please enter your option
3
Enter the name of the metro line to delete a new metro stop (0 for main menu)
MAR
The metro line information is shown below
MAR: Kazlicesme<-4->Yenikapi<-4->Beyazit<-20->Sirkeci<-3->Uskudar<-3->Marmaray
Enter the name of the metro stop to delete (0 for main menu)
Yuénikapi
Metro stop cannot be found.  Enter the name of the metro stop to delete (0 for main menu)
Yenikapi
MAR: Kazlicesme<-8->Beyazit<-20->Sirkeci<-3->Uskudar<-3->Marmaray
```

Execution 3

```
Please enter your option
3
Enter the name of the metro line to delete a new metro stop (0 for main menu)
MAR
The metro line information is shown below
MAR: Kazlicesme<-4->Yenikapi<-4->Beyazit<-20->Sirkeci<-3->Uskudar<-3->Marmaray
Enter the name of the metro stop to delete (0 for main menu)
Kazlicesme
MAR: Yenikapi<-4->Beyazit<-20->Sirkeci<-3->Uskudar<-3->Marmaray
```

4.4 Program Flow & Pathfinder

When executed, your program should ask for the line data file, cost data file and number of the allowed hops, which means the maximum possible transfers between lines and then print the main menu and wait for input.

Execution 3

```
Enter the name of the metro line data file:
lines.txt
Enter the name of costs file:
costs.txt
Enter maximum number of hops:
4
I*********************************************I
I 0 - EXIT PROGRAM                            I
I 1 - PRINT LINES                             I
I 2 - ADD METRO STOP                          I
I 3 - DELETE METRO STOP                       I
I 4 - PATH FINDER                             I
I*********************************************I
>>
Please enter your option
```

Path finder menu item will be used to find a path between two metro stops. Two metro stop names will be requested. If one does not exist, the main menu will appear after an appropriate message (Execution 1). Otherwise, the program will check if the two metro stops are connected within allowed maximum number of hops, if this is true the corresponding path of the best cost will be printed, along with the cost, number of hops and execution time of the path find function (see Execution 2). Otherwise, the user will be informed that such metro line does not exist (Execution 3&4&5). Note that there could be hundreds of paths due to interleaving transfers, and your program should find the one with minimum cost among them. Moreover, adding and deleting metro stops should form new transfers and also may delete them(Execution 6&7). Number of maximum number of hops could change minimum cost path or cause the paths to be unavailable.

Execution 1

```
Please enter your option
4
metroStop1
Where do you want to go?
Taksim
Metro stop does not exist in the table.  Going back to previous menu.
```

Execution 2

```
Where are you now?
Bahcelievler
Where do you want to go?
Tunel
Best cost path:
M1B: Bahcelievler Zeytinburnu
T1:  Zeytinburnu Topkapi Findikzade Sirkeci Karakoy
F2:  Karakoy Tunel
Hops:  2
Cost:  18
Elapsed time:  0.0101102 seconds
```

Execution 3

```
*Maximum hops = 3*
Where are you now?
Ataturk_Havalimani
Where do you want to go?
Uzuncayir
Best cost path:
M1B: Ataturk_Havalimani Atakoy Bahcelievler Zeytinburnu
T1:  Zeytinburnu Topkapi Findikzade Sirkeci
MAR: Sirkeci Uskudar Marmaray
M4:  Marmaray Uzuncayir
Hops:  3
Cost:  40
Elapsed time:  0.00707318 seconds
```

Execution 4

```
*Maximum hops = 1*
Where are you now?
Ataturk_Havalimani
Where do you want to go?
Uzuncayir
Best cost path:
M1B: Ataturk_Havalimani Atakoy
34G: Atakoy Bahcelievler Zeytinburnu Topkapi Edirnekapi Darulaceze Mecidiyekoy Sehitler_Koprusu Uzuncayir
Hops:  1
Cost:  62
Elapsed time:  0.00101842 seconds
```

Execution 5

```
*Maximum hops = 0*
Where are you now?
Ataturk_Havalimani Where do you want to go?
Uzuncayir
No path found between Ataturk_Havalimani and Uzuncayir
Elapsed time:  0.000104805 seconds
```

Execution 6

```
Please enter your option
4
Where are you now?
Taksim
Where do you want to go?
Pendik
Best cost path:
M2:  Taksim Mecidiyekoy
34G: Mecidiyekoy Sehitler_Koprusu Uzuncayir
M4:  Uzuncayir Kozyatagi Maltepe Huzurevi Kartal Pendik
Hops:  2
Cost:  52
Elapsed time:  0.00999671 seconds

>>
Please enter your option
3
Enter the name of the metro line to delete a new metro stop (0 for main menu)
34G
The metro line information is shown below
```

```
34G: Beylikduzu<-5->Guzelyurt<-4->Florya<-4->Atakoy<-3->Bahcelievler<-6->Zeytinburnu<-4->Topkapi<-5->Edirnekapi
<-7->Darulaceze<-21->Mecidiyekoy<-4->Sehitler_Koprusu<-5->Uzuncayir
Enter the name of the metro stop to delete (0 for main menu)
Uzuncayir
34G: Beylikduzu<-5->Guzelyurt<-4->Florya<-4->Atakoy<-3->Bahcelievler<-6->Zeytinburnu<-4->Topkapi<-5->Edirnekapi
<-7->Darulaceze<-21->Mecidiyekoy<-4->Sehitler_Koprusu
>>
Please enter your option
4
Where are you now?
Taksim
Where do you want to go?
Pendik
Best cost path:
F1:  Taksim Kabatas
T1:  Kabatas Karakoy Sirkeci
MAR: Sirkeci Uskudar Marmaray
M4:  Marmaray Uzuncayir Kozyatagi Maltepe Huzurevi Kartal Pendik
Hops:  3
Cost:  57
Elapsed time:  0.00826493 seconds
>>
Please enter your option
2
Enter the name of the metro line to insert a new metro stop (0 for main menu)
M4
The metro line information is shown below
M4:  Kadikoy<-8->Marmaray<-6->Uzuncayir<-7->Kozyatagi<-8->Maltepe<-9->Huzurevi<-11->Kartal<-2->Pendik
Enter the name of the new metro stop
Mecidiyekoy
Enter the name of the previous metro stop for the new one (0 to put the new one as the first metro stop)
Huzurevi
Enter new cost 1
4
Enter new cost 2
11
M4:  Kadikoy<-8->Marmaray<-6->Uzuncayir<-7->Kozyatagi<-8->Maltepe<-9->Huzurevi<-4->Mecidiyekoy<-11->Kartal
<-2->Pendik
>>
Please enter your option
4
Where are you now?
Taksim
Where do you want to go?
Pendik
Best cost path:
M2:  Taksim Mecidiyekoy
M4:  Mecidiyekoy Kartal Pendik
Hops:  1
Cost:  19
Elapsed time:  0.00936298 seconds
```

Note that main menu above each >> is omitted but should be exist in your program.

Although it is enough for your program to work for getting full credit from homework, if you further optimize your algorithm and produce much faster results than the class average, especially for higher hop counts (>20), you may get bonus points from this homework.

Hint: You can use `std::chrono` library to measure your run time. But be careful not to include input waiting time.
Hint: Recursion is a very good way to solve this algorithm. You may use *# of current hops* as exit condition.
Hint: You may determine transfer stops before start searching for the paths.

5 Some Important Rules

In order to get a full credit, your programs must be efficient and well presented, presence of any redundant computation or bad indentation, or missing, irrelevant comments are going to decrease your grades. You also have to use understandable identifier names, informative introduction and prompts. Modularity is also important; you have to use functions wherever needed and appropriate.

When we grade your homeworks we pay attention to these issues. Moreover, in order to observe the real performance of your codes, we may run your programs in *Release* mode and **we may test your programs with very large test cases.**

What and where to submit (PLEASE READ, IMPORTANT): You should prepare (or at least test) your program using MS Visual Studio 2012 or 2019 C++. We will use the standard C++ compiler and libraries of the above mentioned platform while testing your homework. It'd be a good idea to write your name and last name in the program (as a comment line of course). Submissions guidelines are below. Some parts of the grading process are automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade.
Name your cpp file that contains your program as follows:

SUCourseUserName_YourLastname_YourName_HWnumber.cpp

Your SUCourse user name is actually your SUNet username that is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroğlu, then the folder name must be:

cago_Ozbugsizkodyazaroglu_Caglayan_hw1.cpp

Do not add any other character or phrase to the folder name. Make sure that it contains the last version of your homework program. Compress this folder using WINZIP or WINRAR program. Please use "zip" compression. **"rar" or another compression mechanism is NOT allowed.**. Our homework processing system works only with zip files. Therefore, make sure that the resulting compressed file has a zip extension. Check that your compressed file opens up correctly and it contains your cpp file.
You will receive no credits if your compressed folder does not expand or it does not contain the correct files. The name of the zip file should be as follows:

SUCourseUserName_YourLastname_YourName_HWnumber.zip

For example zubzipler_Zipleroglu_Zubeyir_hw1.zip is a valid name, but

hw1_hoz_HasanOz.zip, HasanOzHoz.zip

are **NOT** valid names. **Submit via SUCourse ONLY!** You will receive no credits if you submit by other means (e-mail, paper, etc.).

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck!

CS204 Team (Fatih Taşyaran, Kamer Kaya)