Sabancı University

Faculty of Engineering and Natural Sciences

CS204 Advanced Programming

Fall 2023-2024

Homework
#2

Due: 03/11/2020 - 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!

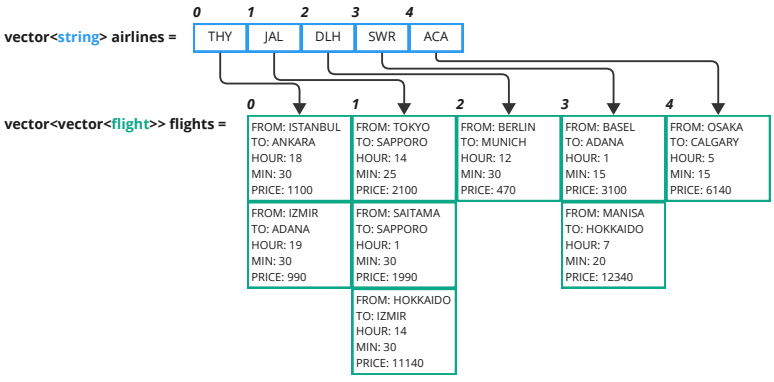
You HAVE TO write down the code on your own.
You CANNOT HELP any friend while coding.
Plagiarism will not be tolerated!!

1 Introduction

The aim of this homework is to use linked lists and doubly linked lists. You will employ these structures to implement an airline passenger's pathfinder program. The number of flights cannot be known; therefore, 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.
Warning: This homework is hard and there will be no deadline extensions. Please start as soon as possible.

2 Program Requirements and Given Functionalities

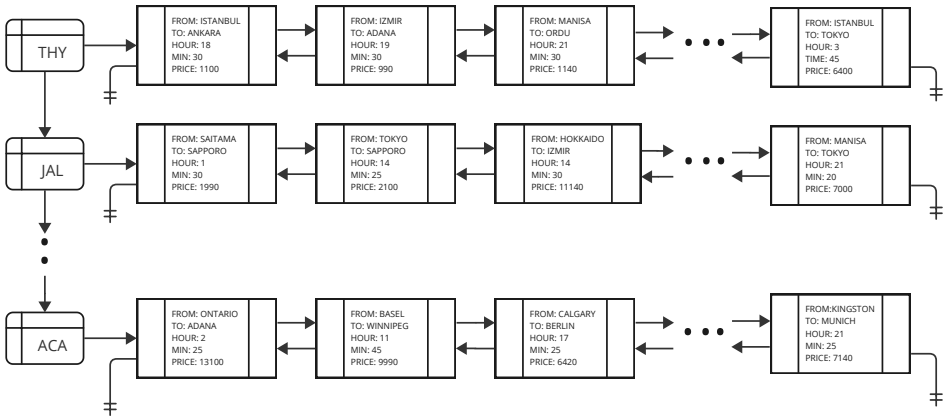
Our passenger is a time traveler, but she is still low on budget. Therefore she wants to optimize her travel costs by searching for the cheapest route while leveraging her ability. You are provided with blocks of code, but you need to implement data structure and search functionalities. You can find the provided code blocks in `assignment.cpp`. In `assignment.cpp`, there are two structs; `airline` and `flights`. You should use those structs as they are for nodes of your linked list structure. But, you need to implement their constructors. Provided file reader returns two vectors; `vector<string> airlines` and `vector<vector<flight>> flights`. You need to construct a linked-list structure from those two vectors. Data is stored in these vectors as the following;



You also provided with the program menu. Program menu automatically pop-up and asks for the next input.

3 Data Structure [40 pts]

You will use a 2D airline and flights list. The airlines will be stored in a single linked list and their flights will be stored in a double linked list. **Flights must be kept ascending in time in the list**. First flight in the list must be the earliest and last flight must be the latest. Here is a visual description of the data structure you need to implement.



Your data structure need to have these functionalities;

- Build the data structure from vectors and keep it always sorted by time.
- Add flights manually. If there is no given airline, create it and append it to the end of the list of airlines.
- Remove flights. If there is no flight left for an airline, remove the airline from the list of airlines.
- Print list. Print the whole data structure in the format that will be given in executions.
- Delete list. Delete whole list and free every pointer allocated with `new`.

4 Pathfinder [60 pts]

You need to implement a pathfinder function that finds the cheapest route with at most requested transfers using the linked list you implemented. You may start from any flight node from any airline in the list that starts from the requested location. Since our passenger is a time traveler, when you take a flight and use it on your route, you can continue with any flight regardless of its time that starts from the same location. However, your route has a maximum number of transfers. You can't report a route that has more transfers than requested. You can find exact examples in executions files.

5 Program Flow

Program flow is determined by the menu that is provided to you. When you select a functionality, it runs the related implementation and prints the menu again until you choose exit. You don't need to modify program flow.

6 Some Remarks

These are the edge cases you **don't need to** consider:

- Linked list is always built from the vectors. You don't need to implement manual addition that operates on empty list.

- `min` property is always a 2 digit number. Day starts with 0 : 10 and ends with 23 : 59
- There will be no two flights that are at the exact same time.
- You don't need to consider travel time or passing day. All flights starts on the same day.
- You don't need to do input check.
- You can use the ordering in input for linked list. You don't need to sort airlines.
- You can implement extra functions and data structures as much as you need. Only restriction is to use the linked list for operations.

Add comments to your code. Your code shouldn't have blocks whose functionality is uncertain.

7 Executions

You can find the test cases in `test_case.txt` files. Menu is omitted from test runs for ease of read.

8 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): We will use the standard C++ compiler and libraries of the 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)