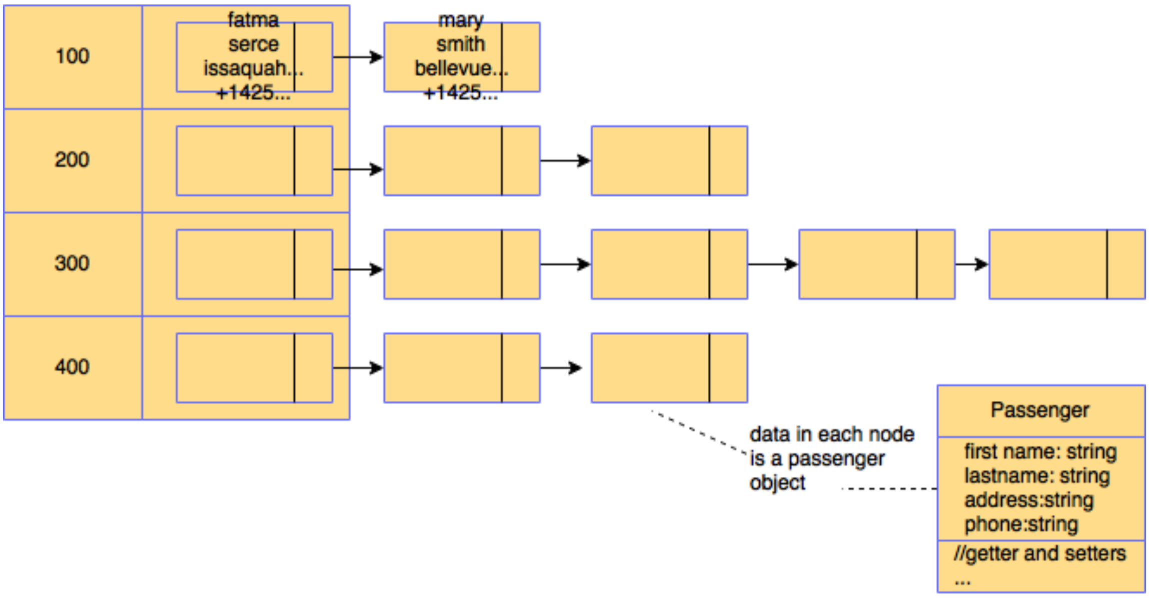
Due Date: October 20, 2017

Delta Airlines maintain four schedule flights per day numbered 100,200,300,400. For each of these flight, they keep a list of passengers as an ordered linked list. The passenger has his/her first name, last name, address, phone number. The database for the airline can be viewed as follows:



The database is required to keep the passengers sorted by their last names in ascending order.

You are asked to write a C++ program that sets up and maintain the database by handling the following commands

ADD, DELETE, SEARCH, LIST, QUIT

Here are the further requirements for the application.

- Create a subclass `OrderedLinkedList.h` which inherits from `LinkedList.h`
- Override `insert`, `delete` and `search` functions in `OrderedLinkedList`. You are allowed to modify `LinkedList.h` if necessary.
- Using the header files `OrderedLinkedList.h` and `LinkedList.h`, make sure the passengers are stored as an ordered linked list: sorted by ascending order using passenger's last name
- Write an application that reads the command from the user and performs the operation
- if the user wants to ADD a new passenger, ask the user to input flight number and also passenger information. Then insert passenger into the appropriate location in the ordered linked list using passenger's last name.
- if the user wants to DELETE a passenger, ask the user to input flight number and also passenger last name. Then remove passenger from the linked list of that flight
- if the user wants to SEARCH a passenger, ask the user to input passenger last name and first name. Then search the user in all flights. If found, print the passenger's flight

number together with his/her info. Assume that first name and last name uniquely identifies the passenger.

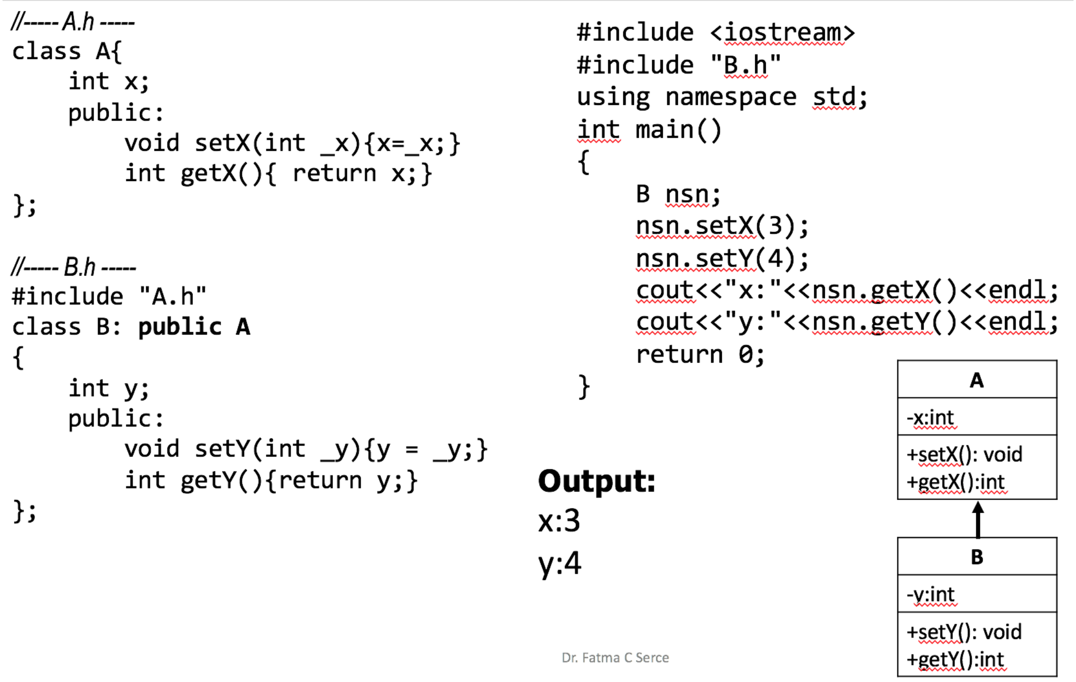
- if the user wants to LIST the passengers, ask the user to input flight number. Then print the flight number and the list of passengers sorted by their last names including all passenger's info.
- Make sure user enters valid input, otherwise prompt an error message
  - Any non-numeric input for flight number is invalid
  - Any numeric input for name and surname is invalid
  - Any value other than 100, 200, 300 and 400 for flight number is invalid
  - There is no validity check required for address and phone!

A sample run:

```
***DELTA AITLINES ***
Please choose an operation:
A(Add) | S (Search) | D(Delete) |L(List) |Q(Quit): A
Enter flight number: 100
Enter first name: MARY
Enter last name: SMITH
Enter address: 12345 NE 50th St Bellevue WA
Enter phone: +14255062396

A(Add) | S (Search) | D(Delete) |L(List) |Q(Quit): A
Enter flight number: 100
Enter first name: FATMA
Enter last name: SERCE
Enter address: 12345 SE 48th St Issaquah WA
Enter phone: +14251234567


A(Add) | S (Search) | D(Delete) |L(List) |Q(Quit): S
Enter last name: SMITH
Enter last name: MARY
Flight Number: 100
First name: MARY
Last name: SMITH
Address: 12345 NE 50th St Bellevue WA
Phone: +14255062396

A(Add) | S (Search) | D(Delete) |L(List) |Q(Quit): L
Enter flight number: 100
FATMA SERCE  [12345 SE 48th St Issaquah WA]   [ +14251234567]
MARY SMITH  [12345 NE 50th St Bellevue WA]   [ +14255062396]
…

A(Add) | S (Search) | D(Delete) |L(List) |Q(Quit): D
```

```
Enter flight number: 100
Enter last name: SMITH
Enter last name: MARY
The passenger is deleted.

A(Add) | S (Search) | D(Delete) |L(List) |Q(Quit): Q
```

Hint: How to implement inheritance and override functions in subclasses in C++?

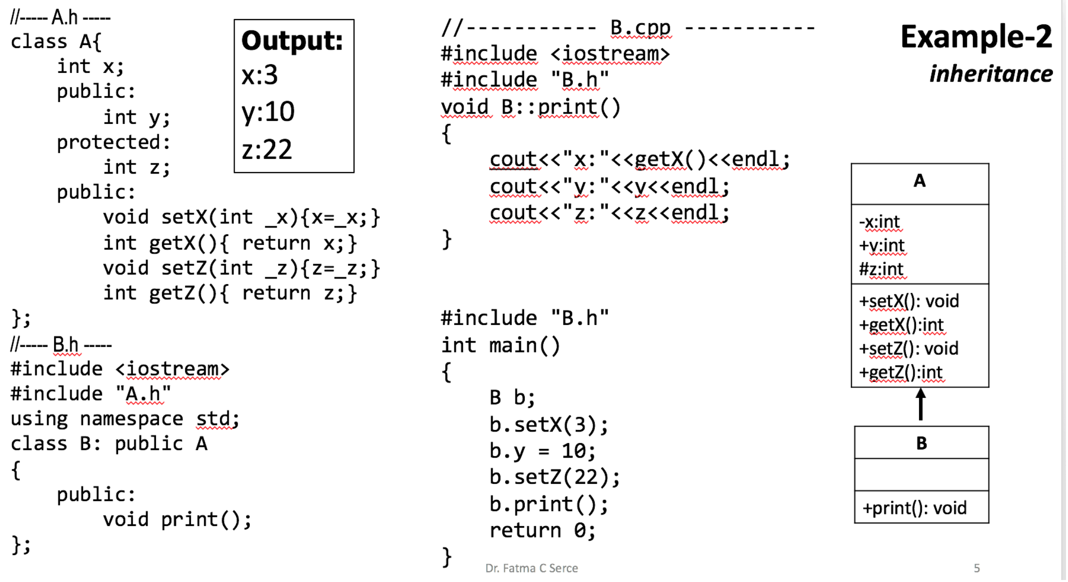


```
//----- A.h -----
class A{
    int x;
    public:
        void setX(int _x){x=_x;}
        int getX(){ return x;}
};

//----- B.h -----
#include "A.h"
class B: public A
{
    int y;
    public:
        void setY(int _y){y = _y;}
        int getY(){return y;}
};
```

```
#include <iostream>
#include "B.h"
using namespace std;
int main()
{
    B nsn;
    nsn.setX(3);
    nsn.setY(4);
    cout<<"x:"<<nsn.getX()<<endl;
    cout<<"y:"<<nsn.getY()<<endl;
    return 0;
}
```

Output:

```
x:3
y:4
```



Example-2
inheritance

```
//----- A.h -----
class A{
    int x;
    public:
        int y;
    protected:
        int z;
    public:
        void setX(int _x){x=_x;}
        int getX(){ return x;}
        void setZ(int _z){z=_z;}
        int getZ(){ return z;}
};
//----- B.h -----
#include <iostream>
#include "A.h"
using namespace std;
class B: public A
{
    public:
        void print();
};
```

Output:

```
x:3
y:10
z:22
```

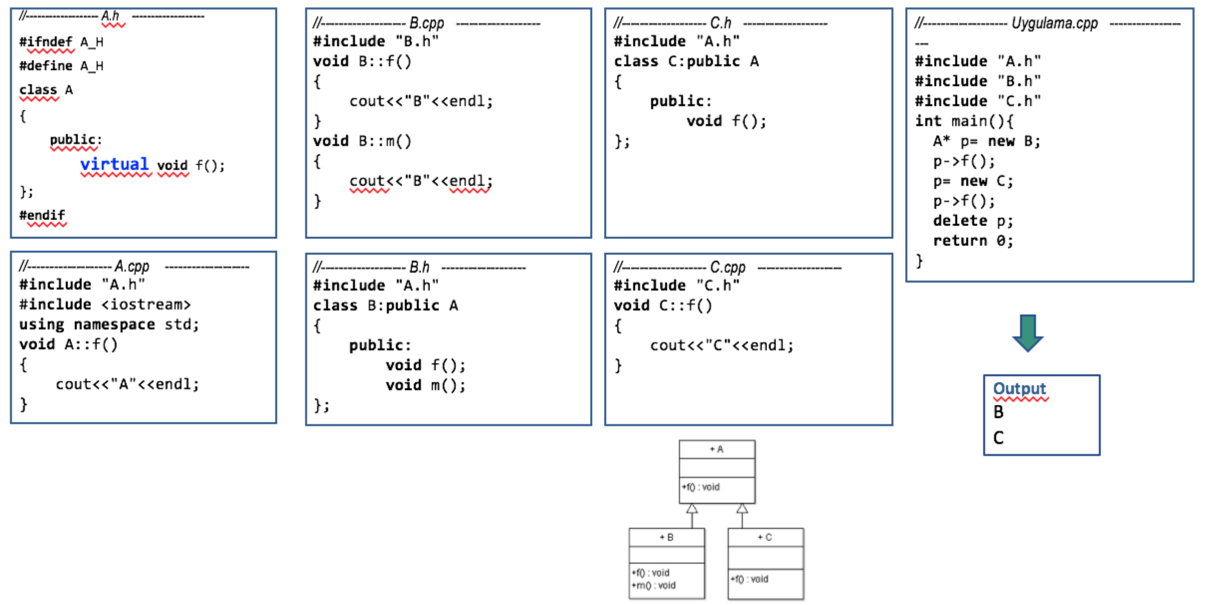
```
//----------- B.cpp -----------
#include <iostream>
#include "B.h"
void B::print()
{
    cout<<"x:"<<getX()<<endl;
    cout<<"y:"<<y<<endl;
    cout<<"z:"<<z<<endl;
}

#include "B.h"
int main()
{
    B b;
    b.setX(3);
    b.y = 10;
    b.setZ(22);
    b.print();
    return 0;
}
```

Virtual Members

- A member of a class that can be redefined in its derived classes is known as a **virtual** member

- In order to declare a member of a class as virtual, we must precede its declaration with the keyword **`virtual`**

```
virtual void f();
```



```
//---------------- A.h ----------------
#ifndef A_H
#define A_H
class A
{
    public:
        virtual void f();
};
#endif
```

```
//---------------- A.cpp ----------------
#include "A.h"
#include <iostream>
using namespace std;
void A::f()
{
    cout<<"A"<<endl;
}
```

```
//---------------- B.cpp ----------------
#include "B.h"
void B::f()
{
    cout<<"B"<<endl;
}
void B::m()
{
    cout<<"B"<<endl;
}
```

```
//---------------- B.h ----------------
#include "A.h"
class B:public A
{
    public:
        void f();
        void m();
};
```

```
//---------------- C.h ----------------
#include "A.h"
class C:public A
{
    public:
        void f();
};
```

```
//---------------- C.cpp ----------------
#include "C.h"
void C::f()
{
    cout<<"C"<<endl;
}
```

```
//---------------- Uygulama.cpp ----------------
--
#include "A.h"
#include "B.h"
#include "C.h"
int main(){
  A* p= new B;
  p->f();
  p= new C;
  p->f();
  delete p;
  return 0;
}
```

```
Output
B
C
```

Submission:

Please follow the following steps for each assignment:

1. Create a separate repository in GitHub, named **`airlinedb`**
2. **Clone** the repository to your local computer.
3. Modify the files and **commit** changes to complete your solution in your local repository.
4. **Push**/sync the changes up to GitHub.
5. To turn in the assignment, send the link for the repository as an email to your instructor (fatma.serce@..)