ELEVATOR SIMULATION

ROLL NO	: CS09B048
NAME	: Aparna K

Problem Statement :
Simulate the 2 elevators over the floors of a building

The following changes have been implemented to improve the old design of the elevator simultor.

Differences from the previous code :

1) Event class has been included as the base class for the events Move_elevator, Person_enters etc. The Event class has virtual function 'happen' which is rewritten in the respective derived classes. This will act as a hot spot and allow any new events to be added in the future easily. The previous code did not have any inheritence for the events.
2) The Passenger class has been made a base class so that different types of passengers can be be realized – for example, VIP, physically challenged etc. These classes derive from the base class. Hence this will serve as a hotspot and will help in future to include any type of person.
3) The capacity can be increased as per the kind of passenger. New algorithms can be implemented for the inclusion of VIPs.
4) The code has been reorganized in .h and .cpp files. Initially it was just in .cpp files. The makefile has been changed to include linking of the .h files.
5) The buiding which was a global object has been made local and is passed as a parameter to the functions by reference. Hence it is no longer common coupling.
6) The code already has the provision to include any number of floors and elevators.

Input format :
For every 4 iterations, a random number of passengers enter the building at various floors with various destinations.

Output format :
For every iteration :
    The passengers added and their source floor.
    The position of the 2 elevators.
    The passengers who enter the elevator in that floors along with waiting time.
    The passengers who leave the elevator at that floor.

Algorithm :
The two elevators always operate in opposite directions and service all requests in its direction. Everytime the top floor or ground floor is reached, the elevator's direction is changed. For heavy crowd, this works well as average waiting time is less.

Cohesion calculation :

floor.h - Passenger object is common for add_passenger and remove_passenger functions.
Building.h – Building object is the parameter for function read.
Event.h – vector of passengers, building object, are common for all functions.

Coupling calculation :
As the parameters are passed by reference, there is stamp coupling.
All other functions have data coupling as parameters are passed by value.