# Влакче

### Общ преглед

Вие управлявате една малка жп гара. Въпреки че гарата е много малка от нея тръгват влакове - както пътнически, така и товарни. Един от коловозите на гарата е зает с вагони за композициите. По другия може да се минава, за да се напусне гарата или пък, за да мине машина, която иска да вземе вагони от другия коловоз.

Вашата задача е по дадена информация за всеки влак да създадете разписание, което спазва правилата за приоритет за преминаване през свободния коловоз.

Приемете, че преминаването по коловоза отнема винаги еднакво време и се случва винаги с еднаква скорост, а след като влакът е преминал по него той вече не може да предизвика конфликт с друг влак.  
  
Приоритетът на преминаване е следният:

Винаги разглеждаме последните **два** добавени влака. От тях влакът се определя по следния приоритет:

1. Товарни влакове с повече от 15 вагона
2. Пътнически влакове
3. Товарни влакове с по-малко от 15 вагона

Ако само един влак е в списък на чакащите, то минава той без значение какъв е

## Подзадача 1: Структура на Влак

### Train

Всички влакове имат номер, име, вид, брой вагони:

Number – цяло число, съдържащо до 5 цифри

Name – текстов низ, без интервали

Type – вид на влака, буква “P” – обозначава пътнически, буква “F” – обозначава товарен

Cars – цяло число, показващо брой вагони на композицията

Реализирайте и конструктор:

* **Train(int number, string name, string type, int cars)** – този конструктор трябва да приема номер на влака, името на влака, видът му и брой вагони. За справка вижте следната схема за Product.cs:

|  |
| --- |
| Train.cs |
| public Train(int number, string name, string type, int cars) {  //TODO: Добавете вашия код тук …  }  //TODO: Добавете ToString() и свойства… |

### Структура за съхранение на чакащите влакове

Трябва да реализирате специална структура, която има два края (**Deque**). Тази структура трябва да позволява добавяне в края и началото, както и извличане на елементите от края и началото, а също така и премахването им. Идеята е пътническите влакове да са в началото, а товарните в края.

|  |
| --- |
| Deque.cs |
| public class Deque<T> : IList<T> {  public Deque() : this(defaultCapacity) {  //празен конструктор, задава капацитета на дека на стойността по подразбиране (16)  }  public Deque(int capacity) {  //създава дека с точно зададен капацитет  }  public Deque(IEnumerable<T> collection)  : this(collection.Count()){  //създава дека с капацитет съответстващ на посочената колекция и прехвърля елементите от колекцията в дека  }  public int Capacity; //показва капацитета  public int Count; //показва броят елементи  public void AddFront(T item) {  //добавя елемент отпред  }  public void AddBack(T item) {  //добавя елемент отзад  }  public T RemoveFront() {  //връща и **премахва** елемента отпред  }  public T RemoveBack() {  //връща и **премахва** елемента отзад  }  public T GetFront() {  //връща, **без да премахва**, елемента отпред  }  public T GetBack() {  //връща, **без да премахва**, елемента отзад  }  } |

## Управление на влаковете

### Kоманди

Вашето приложение трябва да реализира следните команди:

* **Add** **<номер на влак> <име на влак> <вид>** - добавя нов влак, който трябва да премине през коловоза
* **Travel** - разрешава отпътуването на следващия по ред влак. При тази команда, вие трябва да изведете информацията за влака, който отпътува и да го премахнете от информацията с чакащи влакове
* **Next** - при тази команда трябва да отпечатате съобщение “Next train: <номер на влак> <име на влак> <вид>”

## Подзадача 2: Още една команда ☺

Трябва да реализирате ощекоманди:

* **History** - при тази команда трябва да отпечатате информация за вече преминалите влакове по обратния ред на тяхното преминаване, т.е. последният преминал се изпечатва най-напред. Удобно е тази информация да се съхранява в **Stack<T>**

**Забележка:** Освен горепосочените методи трябва да реализирате и необходимите свойства за всеки от класовете. Възможно е да е удачно да реализирате допълнителни полета, свойства и методи, по ваша преценка.

### Скелет

За да се ориентирате по-добре в структурата на решението получавате помощен скелет TrainsSkeleton

### Ограничения

* Броят на влаковете, които може да получите ще бъде не повече от 100
* Номерът на влака ще е цяло число с 5 цифри
* Името на влакът няма да съдържа интервали
* Видът на влака ще се отбелязва с буква - P за пътнически и F за товарен
* Командите Next и History **може** да бъдат извикани дори когато няма информация за чакащи влакове. В такъв случай, вие трябва да игнорирате съответното извикване на командата

## Вход / Изход

### Вход

* Програмата ще получава множество редове с информация. Всеки ред представлява команда. Самият вход се обработва изцяло от примерния Program.cs.
* Всички команди приключват с въвеждането на End

### Изход

За някои от командите не е нужно да извеждате нищо. За останалите формата е:

#### Travel

Ако има влак, който да премине:

{номер} {име} {вид} {вагони}

#### Next

Ако има влак, който да премине:

Next Train: {номер} {име} {вид} {вагони}

#### History

За всеки един от влаковете, които вече са минали изведете по един ред в следния формат:

{номер} {име} {вид} {вагони}

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснение** |
| Add 85611 Karlovo P 2  Add 82223 Sofia P 3  Add 12123 Varna P 4  Next  Travel  Next  Add 12125 Plovdiv P 4  Add 12126 Karnobat P 4  Travel  Next  Travel  Add 12127 Karnobat P 2  End | 12123 Varna P 4  12123 Varna P 4  82223 Sofia P 3  12126 Karnobat P 4  12125 Plovdiv P 4  12125 Plovdiv P 4 | В този пример варненският влак напуска гарата, но след напускането му добавяме два влака – пловдивски и карнобатски, като има точно 2 команди Travel – те придвижват карнобатския и пловдивския влак. След това добавяме и още един карнобатски влак, но понеже преди End командата не извикваме Travel отново – тези влакове не се появяват в изхода |
| Add 85611 Karlovo P 2  Add 82223 Sofia P 3  Add 12123 Varna P 4  Next  Travel  Next  Add 12125 Plovdiv P 4  Add 12126 Karnobat P 4  Travel  History  Next  Travel  History  End | 12123 Varna P 4  12123 Varna P 4  82223 Sofia P 3  12126 Karnobat P 4  12126 Karnobat P 4  12123 Varna P 4  12125 Plovdiv P 4  12125 Plovdiv P 4  12125 Plovdiv P 4  12126 Karnobat P 4  12123 Varna P 4 | В този пример първо заминава варненския влак, после добавяме пловдивски и карнобатски, след което карнобатския заминава. При справката за история изкарваме карнобатския и варненския влак. След това продължаваме с преминаване на пловдивския влак. Отново изакрваме справка, в която фигурират пловдивския, карнобатския и варненския влак – според реда на преминаваме (от най-скорошен към най-отдавнашен) |
| Add 31233 BobovDol F 30  Add 85611 Karlovo P 3  Add 22222 Tulovo F 10  Travel  Travel  Travel  Add 31234 BobovDol F 30  Add 85612 Karlovo P 3  Travel  History  End | 85611 Karlovo P 3  22222 Tulovo F 10  31233 BobovDol F 30  31234 BobovDol F 30  31234 BobovDol F 30  31233 BobovDol F 30  22222 Tulovo F 10  85611 Karlovo P 3 | В този случай първо добавяме 3 влака – товарен за Бобов Дол, пътнически за Карлово и един товарен за Тулово с 10 вагона. Понеже сравняваме само последните 2 влака – пътническият е с приоритет пред товарния, т.като той е с по-малко от 10 вагона – затова карловския минава пръв. След това остават само двата товарни, следователно минава туловския и едва след това този за Бобов Дол.  След това добавяме два влака – товарен за Бобов Дол с 30 вагона, а след него пътнически с 3 вагона – понеже дългите товарни влакове са с приоритет над пътническите – влакът за Бобов Дол минава. В историческата справка излизат двата влака за Бобов Дол, влакът за Тулово и първият влак за Карлово |

## Точки

Разбивката по точки е следната:

1. **30** точки се присъждат, ако вашата програма работи коректно в случаяй, че всички влакове са пътнически. Важно е за тази цел командата **Travel** да работи коректно
2. **30** точки се присъждат, ако вашата програма работи коректно в случай, че влаковете са от какъв да е вид. Важно за тази цел е командата **Travel** и командата **Next** да работят коректно
3. **40** точки се присъждат, ако вашата програма отпечатва коректно историята на преминалите влакове

Общ брой точки: **100**