

# Języki Skryptowe

Dokumentacja projektu „Task Railways”

Dominik Szczepanik, grupa 2/4

Informatyka - Wydział Matematyki Stosowanej - semestr III

5 styczeń 2023

# Część I

## Opis programu

Problem, który rozwiązuje program to zarządzanie systemem rezerwacji miejsc dla pojedynczej linii kolejowej InterCity.

Dla uproszczenia przyjmujemy, że połączenie InterCity przebiega przez  $n$  miast ponumerowanych kolejno od 1 do  $n$  (miasto na początku trasy ma numer 1, a na końcu  $n$ ). W pociągu jest  $m$  miejsc i między żadnymi dwiema kolejnymi stacjami nie można przewieźć większej liczby pasażerów.

System informatyczny ma przyjmować kolejne zgłoszenia i stwierdzać, czy można je zrealizować. Zgłoszenie jest akceptowane, gdy na danym odcinku trasy w pociągu jest wystarczająca liczba wolnych miejsc, w przeciwnym przypadku zgłoszenie jest odrzucane. Nie jest możliwe częściowe zaakceptowanie zgłoszenia, np. na część trasy, albo dla mniejszej liczby pasażerów. Po zaakceptowaniu zgłoszenia uaktualniany jest stan wolnych miejsc w pociągu. Zgłoszenia przetwarzane są jedno po drugim w kolejności nadchodzenia.

## Wejście

W pierwszym wierszu standardowego wejścia znajdują się trzy liczby całkowite  $n$ ,  $m$  i  $z$  ( $1 \leq n \leq 60\,000$ ,  $1 \leq m \leq 60\,000$ ,  $1 \leq z \leq 60\,000$ ) pooddzielane pojedynczymi odstępami, oznaczające odpowiednio: liczbę miast na trasie, liczbę miejsc w pociągu i liczbę zgłoszeń. W kolejnych wierszach opisane są kolejne zgłoszenia. W wierszu o numerze  $i+1$  opisane jest  $i$ -te zgłoszenie. Zapisane są w nim trzy liczby całkowite  $p$ ,  $k$  i  $L$  ( $1 \leq p < k \leq n$ ,  $1 \leq L \leq m$ ) pooddzielane pojedynczymi odstępami, oznaczające odpowiednio: numer stacji początkowej, numer stacji docelowej i wymaganą liczbę miejsc.

## Wyjście

Program wypisuje na standardowe wyjście kolejne wiersze z odpowiedzią na zgłoszenie T-tak N-nie.

## Przykład

Dla danych wejściowych:

4 6 4

1 4 2

1 3 2

2 4 3

1 2 3

poprawną odpowiedzią jest:

T

T

N

N

## Instrukcja obsługi

Aby uruchomić program należy włączyć skrypt menu.bat otwierający menu obsługi naszego programu. Po uruchomieniu wyświetli nam się menu z czterema opcjami, które są ponumerowane od 1 do 4. Użytkownik powinien wpisać numer opcji którą wybiera.

```
Co chcesz zrobić?  
1 - Uruchom program  
2 - Informacje o projekcie  
3 - Backup  
4 - Wyjście  
wybór:
```

Rysunek 1: Menu programu

Możliwe wybory są następujące:

1. Uruchom program – uruchamia skrypt pythona który pobiera wszystkie dane z katalogu „inputs”, a następnie wyniki zapisuje do folderu „outputs”. Uruchamiany jest również drugi skrypt pythona który tworzy plik „raport.html” podsumowujący uzyskane informacje. Ostatecznie plik html jest wyświetlany w domyślnej przeglądarce internetowej.

```
Co chcesz zrobić?  
1 - Uruchom program  
2 - Informacje o projekcie  
3 - Backup  
4 - Wyjście  
wybór:  
1  
skrypt został uruchomiony, wyniki zostaną wyświetlone w domyś  
lnej przeglądarce, możesz je również przejrzeć otwierając pli  
k raport.html  
aby powrócić do menu naciśnij dowolny klawisz na klawiaturze.  
.  
Press any key to continue . . .
```

Rysunek 2: komunikat o pomyślnej próbie uruchomienia programu

## Raport z działania programu Task Railways | 2023-01-05 20:08:19.909657

Inputs	Outputs
4 6 4	
1 4 2	T
1 3 2	T
2 4 3	N
1 2 3	N
7 20 5	
1 7 10	T
2 4 5	T
3 4 6	N
4 5 10	T
6 7 1	T
10 1000 11	
1 2 200	T
1 3 400	T
1 5 100	T
1 10 200	T
3 7 100	T
4 5 500	N
2 4 400	N
3 6 400	T
6 10 100	T
9 10 500	T
9 10 20	T

Rysunek 3: Przykładowy raport programu

2. Informacje o projekcie - wypisuje na ekranie konsoli opis założeń programu

```
wybór:
2
Autor projektu: Dominik Szczepanik
-----
Problem który rozwiązuje program to zarządzanie systemem reze
rwacji miejsc dla pojedynczej linii kolejowej InterCity.

Dla uproszczenia przyjmujemy, że połączenie InterCity przebie
ga przez n miast ponumerowanych kolejno od 1 do n (miasto na
początku trasy ma numer 1, a na końcu n). W pociągu jest m mi
ejsc i między żadnymi dwiema kolejnymi stacjami nie można prz
ewieźć większej liczby pasażerów.

System informatyczny ma przyjmować kolejne zgłoszenia i stwie
rdzać, czy można je zrealizować. Zgłoszenie jest akceptowane,
gdy na danym odcinku trasy w pociągu jest wystarczająca liczb
a wolnych miejsc, w przeciwnym przypadku zgłoszenie jest odr
zucane. Nie jest możliwe częściowe zaakceptowanie zgłoszenia,
np. na część trasy, albo dla mniejszej liczby pasażerów. Po
zaakceptowaniu zgłoszenia uaktualniany jest stan wolnych miej
sc w pociągu. Zgłoszenia przetwarzane są jedno po drugim w ko
lejności nadchodzenia.

program wypisuje na standardowe wyjście wiersze z odpowiedzią
kolejno na n-te zgłoszenie T-tak N-nie.
-----

naciśnij dowolny klawisz by wrócić do menu głównego.
```

Rysunek 4: Informacje o projekcie

3. Backup - tworzy kopię zapasową danych w katalogu Backups zawierającą raport.html oraz zawartość folderów inputs i outputs

```
wybór:
3
inputs\input1.txt
inputs\input2.txt
inputs\input3.txt
3 File(s) copied
outputs\output1.txt
outputs\output2.txt
outputs\output3.txt
3 File(s) copied
C:raport.html
1 File(s) copied
backup został wykonany pomyślnie, naciśnij dowolny klawisz aby
powrócić do menu głównego
Press any key to continue . . .
```

Rysunek 5: komunikat o pomyślnym wykonaniu backup 'a

```
\---2023_01_05__20_38_03
|   raport.html
|
+---inputs
|   input1.txt
|   input2.txt
|   input3.txt
|
\---outputs
    output1.txt
    output2.txt
    output3.txt
```

Rysunek 6: Przykładowa struktura backupu

#### 4. Wyjście - zamyka menu

## Struktura danych programu

Program składa się z następującej struktury danych, wymaganych do prawidłowego uruchomienia aplikacji:

**menu.bat** - skrypt batch będący menu, który uruchamia program, wyświetla informacje o programie jak i tworzy kopie zapasową danych otrzymanych w wyniku wykonania tego programu

**main.py** - skrypt python zawierający główny program, pobiera on pliki wejściowe z katalogu „inputs”, przetwarza je, a następnie rezultat obliczeń zapisuje do folderu „outputs”

**raport.py** - Skrypt python pobierający dane z folderu „inputs” oraz „outputs” i generujący plik raport.html zawierający raport wszystkich danych w postaci tabeli

**info.bat** – plik batch wypisujący na ekranie konsoli opis założeń programu

**style.css** – plik css który odpowiada za stronę stylistyczną „raport.html”

**folder „inputs”** – folder zawierający pliki wejściowe programu

Program w wyniku działania tworzy katalogi „outputs”, „Backups” oraz plik „raport.html”, które nie są wymagane do prawidłowego uruchomienia aplikacji.

**folder „outputs”** – folder zawierające pliki wyjściowe programu zawierające rezultat działania programu.

**folder „Backups”** – folder kopii zapasowej danych zawierającej raport.html oraz zawartość folderów inputs i outputs

**raport.html** – plik html zawierający raport wszystkich danych w postaci tabeli, który możemy podejrzeć w przeglądarce internetowej

```

|   info.bat
|   main.py
|   menu.bat
|   raport.html
|   raport.py
|   style.css
|
+---Backups
|   \---2023_01_05__20_38_03
|       |   raport.html
|       |
|       +---inputs
|       |   |   input1.txt
|       |   |   input2.txt
|       |   |   input3.txt
|       |   |
|       |   \---outputs
|       |       |   output1.txt
|       |       |   output2.txt
|       |       |   output3.txt
|       |
|       +---inputs
|       |   |   input1.txt
|       |   |   input2.txt
|       |   |   input3.txt
|       |
|       \---outputs
|           |   output1.txt
|           |   output2.txt
|           |   output3.txt

```

Rysunek 7: Struktura plików projektu w formie drzewa



## Część II

### Opis działania

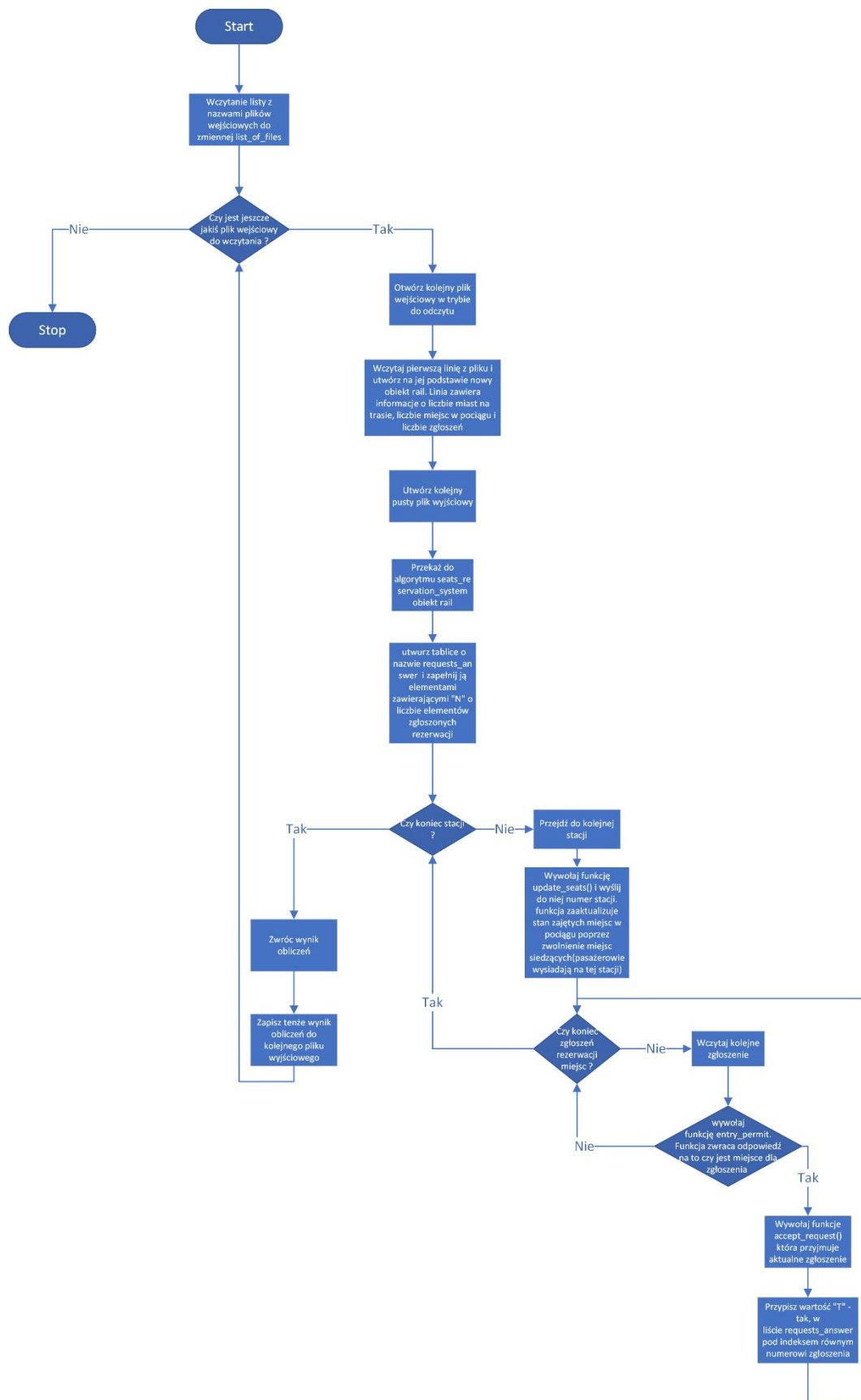
Skrypt menu.bat po wybraniu opcji „uruchom program”, uruchamia skrypt main.py, który pobiera wszystkie nazwy plików z katalogu „inputs”. Następnie program w pętli otwiera kolejne pliki wejściowe i czytuje z nich każdą linię. Pierwsza czytana linia z każdego pliku wejściowego oznacza kolejno w kolumnach liczbę miast na trasie, liczbę miejsc w pociągu i liczbę zgłoszeń. Następne wczytywane linie to zgłoszenia, które zawierają kolejno w kolumnach numer stacji początkowej, numer stacji docelowej i wymaganą liczbę miejsc.

Wczytane informacje przekazywane są dalej do algorytmu, który sprawdza wszystkie zgłoszenia i decyduje o ich przyjęciu czy też odrzuceniu. Rezultat zapisywany jest do kolejnych plików wyjściowych w folderze „outputs”.

Otrzymane w ten sposób wyniki są następnie przetwarzane przez raport.py, który tworzy plik raport.html, który zaś w tabeli umieszcza zarówno zawartość plików wejściowych oraz wyjściowych.

Na końcu w domyślnej przeglądarce internetowej otwierany jest utworzony wcześniej plik „raport.html”.

# Algorytm



Rysunek 8: Algorytm programu w postaci schematu blokowego

## Wykorzystane biblioteki i przykłady ich użycia

```
import os
```

```
list_of_files = os.listdir(folder_path)
```

Rysunek 9: użycie biblioteki **os** w main.py

```
import datetime
import os

list_of_input_files = os.listdir("inputs")
list_of_output_files = os.listdir("outputs")

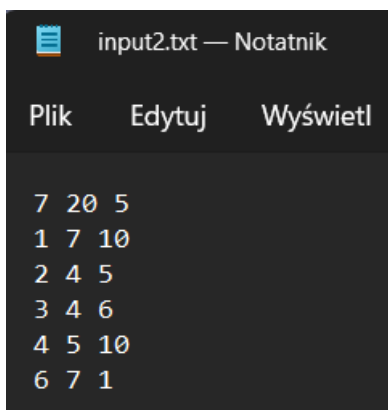
inputs_content = []
outputs_content = []

current_date = datetime.datetime.now()
```

Rysunek 10: użycie biblioteki **os** i **datetime** w raport.py

## Testy

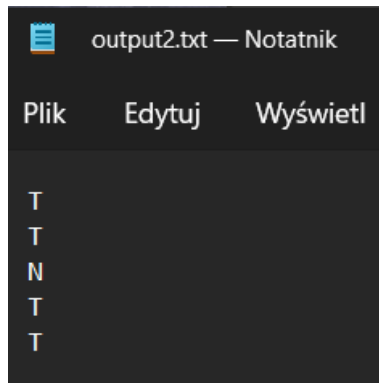
Dane wejściowe:



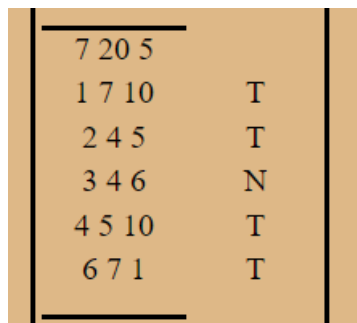
Plik	Edytuj	Wyświetl
7	20	5
1	7	10
2	4	5
3	4	6
4	5	10
6	7	1

Rysunek 11: zawartość pliku input2.txt

Dane wyjściowe:



Rysunek 12: zawartość pliku output2.txt



7	20	5	
1	7	10	T
2	4	5	T
3	4	6	N
4	5	10	T
6	7	1	T

Rysunek 13: wynik zapisany do raportu

## Pełen kod aplikacji

### main.py

```
import os

class Rail:

    def __init__(self, num_of_cities, sitting_places, requests, file):

        if num_of_cities < 1 or num_of_cities > 60000:
            raise Exception("Incorrect amount of cities")
        self.__num_of_cities = num_of_cities

        if sitting_places < 1 or sitting_places > 60000:
            raise Exception("Incorrect amount of sitting places")
        self.__sitting_places = sitting_places

        if requests < 1 or requests > 60000:
            raise Exception("Incorrect amount of requests")
        self.__requests = requests
```

```

        # destinations includes starting station, destination station,
number of passengers
        self.__destinations = []

        self.__people_inside = 0

        self.__requests_in_progress = []

        self.__file = file

    def get_number_of_cities(self):
        return self.__num_of_cities

    def get_number_of_requests(self):
        return self.__requests

    # adding all requests to database
    def add_requests(self):
        for i in range(self.__requests):

            request = [int(e) for e in self.__file.readline().split(" ")]

            # validation of request
            if len(request) != 3:
                raise Exception("Incorrect length of request")
            else:
                if request[0] < 1 or request[0] >= request[1]:
                    raise Exception("Incorrect start station")
                elif request[1] > self.__num_of_cities:
                    raise Exception("Incorrect destination station")
                elif 1 > request[2] or request[2] > self.__sitting_places:
                    raise Exception("Incorrect numbers of requests for
seats")

            # adding single request to database
            self.__destinations.append(request)

    def entry_permit(self, request, current_station):
        if self.__destinations[request][0] == current_station and
self.__destinations[request][2]+self.__people_inside \
        <= self.__sitting_places:
            return True
        else:
            return False

    def accept_request(self, request):
        self.__people_inside += self.__destinations[request][2]
        self.__requests_in_progress.append(request)

    def update_seats(self, current_station):
        for e in self.__requests_in_progress:
            if self.__destinations[e][1] == current_station:
                self.__people_inside -= self.__destinations[e][2]

def seats_reservation_system(rail):
    requests_answer = ["N"] * rail.get_number_of_requests()

    rail.add_requests()

    for s in range(1, rail.get_number_of_cities()+1):

```

```

        rail.update_seats(s)
        for r in range(rail.get_number_of_requests()):
            if rail.entry_permit(r, s):
                rail.accept_request(r)
                requests_answer[r] = "T"

    result_output = ""
    for i in requests_answer:
        result_output += i+"\n"
    return result_output

folder_path = "inputs"

list_of_files = os.listdir(folder_path)

count_file = 1
for file_name in list_of_files:
    with open("inputs/"+file_name, 'r') as f:
        f_content = f.readline()
        inf = [int(e) for e in f_content.split(" ")]
        rail_1 = Rail(inf[0], inf[1], inf[2], f)

        with open("outputs/output"+str(count_file)+".txt", 'w') as
output_file:
            output_file.write(seats_reservation_system(rail_1))

        count_file += 1

```

---

## raport.py

```

import datetime
import os

list_of_input_files = os.listdir("inputs")
list_of_output_files = os.listdir("outputs")

inputs_content = []
outputs_content = []

current_date = datetime.datetime.now()

for file_name in list_of_input_files:
    with open("inputs/"+file_name) as input_file:
        temp_list = []
        for line in input_file:
            temp_list.append(line.rstrip())
        inputs_content.append(temp_list)

```

```

for file_name in list_of_output_files:
    with open("outputs/"+file_name) as output_file:
        temp_list = []
        for line in output_file:
            temp_list.append(line.rstrip())
        outputs_content.append(temp_list)

html_content = '''<html> <head><link rel="stylesheet" href="style.css">
</head> <body><h3>Raport z dzialania programu
Task Railways |
''' + str(current_date) + '''</h3><table><tr><th>Inputs</th><th>Outputs</th></tr>'''

for idx1, file_content in enumerate(inputs_content):
    for idx2, row in enumerate(file_content):
        if idx2 == 0:
            html_content += "<tr><td><div>" + row + "</div></td></tr>"
        else:
            html_content +=
"<tr><td>" + row + "</td><td>" + outputs_content[idx1][idx2-1] + "</td></tr>"

html_content += "</table></body></html>"

with open("raport.html", "w") as html_file:
    html_file.write(html_content)

```

---

## menu.bat

```
@echo off & @chcp 1250>nul
```

```
:start
```

```
cls
```

```
echo Co chcesz zrobić?
```

```
echo 1 - Uruchom program
```

```
echo 2 - Informacje o projekcie
```

```
echo 3 - Backup
```

```
echo 4 - Wyjscie
```

```
echo wybór:
```

```
set /p whatapp=
```

```
if %whatapp%==1 (  
    goto 1  
) else if %whatapp%==2 (  
    goto 2  
) else if %whatapp%==3 (  
    goto 3  
) else if %whatapp%==4 (  
    goto 4  
)
```

```
:1
```

```
cmd /c python main.py
```

```
cmd /c python raport.py
```

```
cmd /c start raport.html
```

```
echo skrypt został uruchomiony, wyniki zostaną wyświetlone w domyślnej  
przeglądarce, możesz je również przejrzeć otwierając plik raport.html
```

```
echo aby powrócić do menu naciśnij dowolny klawisz na klawiaturze..
```

```
pause
```

```
goto start
```

```
:2
```

```
info.bat
```

```
goto start
```

```
:3
```

```
for /f %%a in ('powershell -Command "Get-Date -format  
yyyy_MM_dd_HH_mm_ss"') do set datetime=%%a
```

```
if not exist "Backups" mkdir Backups
```

```
cd Backups
```



```
mkdir "%datetime%"
cd %datetime%
mkdir inputs
mkdir outputs
cd ../
cd ../
xcopy /s inputs Backups\ "%datetime%" \inputs
xcopy /s outputs Backups\ "%datetime%" \outputs
xcopy raport.html Backups\ "%datetime%"
echo backup został wykonany pomyślnie, naciśnij dowolny klawisz aby powrócić
do menu głównego
pause
goto start

:4
```

---

## info.bat

```
@echo off
```

```
echo Autor projektu: Dominik Szczepanik
```

```
echo -----
```

```
echo Problem który rozwiązuje program to zarządzanie systemem rezerwacji
miejsc dla pojedynczej linii kolejowej InterCity.
```

```
echo.
```

```
echo Dla uproszczenia przyjmujemy, że połączenie InterCity przebiega przez n
miast ponumerowanych kolejno od 1 do n (miasto na początku trasy ma numer 1,
a na końcu n). W pociągu jest m miejsc i między żadnymi dwiema kolejnymi
stacjami nie można przewieźć większej liczby pasażerów.
```

```
echo.
```

echo System informatyczny ma przyjmować kolejne zgłoszenia i stwierdzać, czy można je zrealizować. Zgłoszenie jest akceptowane, gdy na danym odcinku trasy w pociągu jest wystarczająca liczba wolnych miejsc, w przeciwnym przypadku zgłoszenie jest odrzucane. Nie jest możliwe częściowe zaakceptowanie zgłoszenia, np. na część trasy, albo dla mniejszej liczby pasażerów. Po zaakceptowaniu zgłoszenia uaktualniany jest stan wolnych miejsc w pociągu. Zgłoszenia przetwarzane są jedno po drugim w kolejności nadchodzenia.

echo.

echo program wypisuje na standardowe wyjście wiersze z odpowiedzią kolejno na n-te zgłoszenie T-tak N-nie.

echo -----

echo.

echo naciśnij dowolny klawisz by wrócić do menu głównego.

pause

menu.bat