

Penjabaran Masalah

1. Elemen Komputasi

a. Operasi matematika, untuk menaikkan indeks dalam perulangan, menambahkan nilai dari variabel lain dan membuat operasi perkalian.

```
1 amountOfGoods + 1
```

```
1 totalWeight += weight
2 totalValue += priority
```

```
1 weightTaken = fraction * weight
```

b. Operasi String, untuk manipulasi string (menggabungkan/mengubah huruf besar kecil/memformat string) dan proses input pengguna.

```
1 trainClass = input("Select Class : ").capitalize()
```

```
1 if input("\nWant to try another class? (Yes/No): ").lower() != 'yes':
```

```
1 f"\nThis is the travel route from {originStation} to {destinationStation}: \n{' -> '.join(track)} \nDuration: {routeDuration} hours")
```

```
1 if trainClassInput.lower() in ['economy', 'business', 'exclusive']:
```

```
1 f"Total cost : ${totalCost}"
```

c. Pernyataan kondisi, untuk memproses program berdasarkan kondisi tertentu.

```
1 if station in visited:
```

```
1 if goodsCarried == "no":
```

```
1 if not selectedTrainClass:
```

d. Melakukan Perulangan, untuk mengulangi program yang sama dalam kondisi benar.

```
1 for i in range
```

```
1 while True:
```

2. Elemen Dekomposisi

a. Fungsi, untuk membuat program satu kesatuan dan membaginya menjadi sub-sub kecil.

```
1 def dijkstra(route, begin, end):
```

```
1 def viewRoutes():
```

```
1 def fractionalKnapsack(selectedTrainClass, items):
```

```
1 def oneZeroKnapsack(selectedTrainClass, items):
```

```
1 def viewTrain():
```

b. Struktur data, untuk menyimpan informasi yang mewakili detail program.

```
1 itemRecommendations
```

```
1 table = []
```

```
1 station
```

3. Elemen Pattern Recognition

a. Pencocokan pola, untuk memeriksa apakah nilai dalam suatu variabel cocok dengan pencarian.

```
1 if choice == '1':
```

```
1 if confirmation.lower() == 'yes':
```

```
1 if labelFr == "whole":
```

```
1 if station == end:
```

4. Elemen Generalisasi

a. Struktur data generik dan desain yang modular, untuk menyimpan data sebagai kamus dan membuatnya lebih mudah beradaptasi dengan program lain tanpa modifikasi/pembuatan program terbaru.

```

1 def dijkstra(route, begin, end):
2     queue = [(0, begin, [])]
3     visited = set()
4     while queue:
5         routeDuration, station, track = heapq.heappop(queue)
6         if station in visited:
7             continue
8         visited.add(station)
9         track = track + [station]
10        if station == end:
11            return routeDuration, track
12        for nextStation, distance in route.get(station, {}).items():
13            if nextStation not in visited:
14                heapq.heappush(
15                    queue, (routeDuration + distance, nextStation, track))
16    return float("inf"), []

```

```

1 train = [
2     {'trainClassName': 'Economy',
3      'trainCapacity': 25, 'trainClassPrices': 20},
4     {'trainClassName': 'Business',
5      'trainCapacity': 30, 'trainClassPrices': 30},
6     {'trainClassName': 'Exclusive',
7      'trainCapacity': 35, 'trainClassPrices': 40}
8 ]

```

```

1 route = {
2     "Minstowe": {"Cowstone": 3},
3     "Oldcastle": {"New North": 5, "Freeham": 2},
4     "Cowstone": {"Minstowe": 3, "New North": 4, "Bingborough": 6, "Donningpool": 7, "Highbrook": 5, "Freeham": 2},
5     "New North": {"Oldcastle": 5, "Cowstone": 4, "Bingborough": 3, "Donningpool": 6, "Wington": 4, "Highbrook": 2},
6     "Freeham": {"Oldcastle": 2, "Cowstone": 2, "Donningpool": 3, "Wington": 5},
7     "Bingborough": {"Cowstone": 6, "New North": 3, "Donningpool": 2, "Highbrook": 1},
8     "Donningpool": {"Cowstone": 7, "New North": 6, "Freeham": 3, "Bingborough": 2, "Wington": 4, "Highbrook": 5, "Old Mere": 2},
9     "Highbrook": {"Cowstone": 5, "New North": 2, "Bingborough": 1, "Donningpool": 5},
10    "Wington": {"New North": 4, "Freeham": 5, "Donningpool": 4},
11    "Old Mere": {"Donningpool": 2}
12 }
13

```


Kode Program

```

1 import heapq
2 import random
3 from datetime import datetime, timedelta
4 # jika ingin menggunakan kode program ini, pastikan sudah menginstall "pip install tabulate" di terminal.
5 from tabulate import tabulate
6 # import fraction ini, bukan berarti langsung membagi menjadi pecahan. namun merubah bilangan desimal (0,5) menjadi pecahan (1/2).
7 from fractions import Fraction
8
9 stations = ["Minstowe", "Oldcastle", "Cowstone",
10            "New North", "Freeham", "Bingborough", "Donningpool", "Old Mere", "Highbrook", "Wington"]
11 route = {
12     "Minstowe": {"Cowstone": 3},
13     "Oldcastle": {"New North": 5, "Freeham": 2},
14     "Cowstone": {"Minstowe": 3, "New North": 4, "Bingborough": 6, "Donningpool": 7, "Highbrook": 5, "Freeham": 2},
15     "New North": {"Oldcastle": 5, "Cowstone": 4, "Bingborough": 3, "Donningpool": 6, "Wington": 4, "Highbrook": 2},
16     "Freeham": {"Oldcastle": 2, "Cowstone": 2, "Donningpool": 3, "Wington": 5},
17     "Bingborough": {"Cowstone": 6, "New North": 3, "Donningpool": 2, "Highbrook": 1},
18     "Donningpool": {"Cowstone": 7, "New North": 6, "Freeham": 3, "Bingborough": 2, "Wington": 4, "Highbrook": 5, "Old Mere": 2},
19     "Highbrook": {"Cowstone": 5, "New North": 2, "Bingborough": 1, "Donningpool": 5},
20     "Wington": {"New North": 4, "Freeham": 5, "Donningpool": 4},
21     "Old Mere": {"Donningpool": 2}
22 }
23
24 train = [
25     {'trainClassName': 'Economy',
26      'trainCapacity': 25, 'trainClassPrices': 20},
27     {'trainClassName': 'Business',
28      'trainCapacity': 30, 'trainClassPrices': 30},
29     {'trainClassName': 'Exclusive',
30      'trainCapacity': 35, 'trainClassPrices': 40}
31 ]

```

1. Menu Utama



```
1 def mainMenu():
2     print("=====")
3     print("Welcome to the Train ticket booking program")
4     name = input("\nBefore using the program, enter your first name: ")
5     print("=====")
6     while True:
7         print(f"\nWelcome to {name} Express")
8         print("=====")
9         print("1. Order Ticket")
10        print("2. View Routes")
11        print("3. View Train")
12        print("4. Exit")
13        print("=====")
14        choice = input("choose (1-4): ")
15
16        if choice == '1':
17            orderTicket()
18        elif choice == '2':
19            viewRoutes()
20        elif choice == '3':
21            viewTrain()
22        elif choice == '4':
23            print("Thank you for using our program.")
24            break
25        else:
26            print("choose is invalid. Please try again.")
27
28
29 mainMenu()
```

2. Penentuan Rute

```

1 def dijkstra(route, begin, end):
2     queue = [(0, begin, [])]
3     visited = set()
4     while queue:
5         routeDuration, station, track = heapq.heappop(queue)
6         if station in visited:
7             continue
8         visited.add(station)
9         track = track + [station]
10        if station == end:
11            return routeDuration, track
12        for nextStation, distance in route.get(station, {}).items():
13            if nextStation not in visited:
14                heapq.heappush(
15                    queue, (routeDuration + distance, nextStation, track))
16    return float("inf"), []

```

```

1 def viewRoutes():
2     stations = list(route.keys())
3     displayed = set()
4     table = []
5     for begin in stations:
6         for end in route[begin]:
7             if (begin, end) not in displayed and (end, begin) not in displayed:
8                 routeDuration = route[begin][end]
9                 table.append([f"{begin} -- {end}", routeDuration])
10                displayed.add((begin, end))
11    print("\nCost : $15/jam")
12    print(tabulate(table, headers=[
13        "Routes (2-way)", "Duration (Hours)", tablefmt="outline"])
14
15    while True:
16        print("Enter your route")
17        print("=====")
18        originStation = input("\nFrom : ")
19        destinationStation = input("To   : ")
20        if originStation not in stations or destinationStation not in stations:
21            print("Invalid station. Please try again.")
22            continue
23
24        routeDuration, track = dijkstra(
25            route, originStation, destinationStation)
26        if routeDuration == float("inf"):
27            print("No routes available.")
28        else:
29            print(
30                f"This is the shortest route from {originStation} to {destinationStation}: \n{' -> '.join(track)} \nDuration: {routeDuration} hours")
31
32        if input("\nSee another route? (Yes/No): ").lower() != 'yes':
33            break

```

3. Penentuan Kereta Api

```

1  def viewTrain():
2      while True:
3          table = []
4          for trainn in train:
5              table.append([trainn['trainClassName'],
6                           trainn['trainCapacity'], trainn['trainClassPrices']])
7          print(tabulate(table, headers=[
8              'Train Name', 'Capacity (kg)', 'Price ($) per kg'], tablefmt="grid"))
9          goodsCarried = input(
10             "we can help you choose what to bring, whether you want to try ?\nyes/no = ")
11         if goodsCarried == "no":
12             while True:
13                 trainClass = input("Select Class : ").capitalize()
14                 if trainClass in ['Economy', 'Business', 'Exclusive']:
15                     break
16                 else:
17                     print(
18                         "The train class you selected is invalid. Please try again.")
19
20                 selectedTrainClass = next(
21                     (trainn for trainn in train if trainn['trainClassName'] == trainClass), None)
22
23                 if not selectedTrainClass:
24                     print("The train class you selected is not available.")
25                     return
26
27                 selectedTrainClass = selectedTrainClass.copy()
28                 break
29
30         elif goodsCarried not in ["no", "yes"]:
31             print("you entered it wrong. try again")
32             viewTrain()
33
34         while True:
35             trainClass = input("Select Class : ").capitalize()
36             if trainClass in ['Economy', 'Business', 'Exclusive']:
37                 break
38             else:
39                 print("The train class you selected is invalid. Please try again.")
40
41             selectedTrainClass = next(
42                 (trainn for trainn in train if trainn['trainClassName'] == trainClass), None)
43
44             if not selectedTrainClass:
45                 print("The train class you selected is not available.")
46                 return
47
48             selectedTrainClass = selectedTrainClass.copy()
49
50         if goodsCarried == "yes":
51             items = []
52             print(
53                 "Give your items a priority scale ranging from 1 (very important) to 5 (not important)")
54             amountOfGoods = int(
55                 input("How many items do you want to bring? "))
56
57             for i in range(1, amountOfGoods + 1):
58                 print(f"\nItem-{i}")
59                 name = input(f"Name of Item-{i}\t\t\t: ")
60                 weight = float(input(f"Weight of Item-{i} (kg)\t\t: "))
61                 priority = int(input(f"Priority Of Item-{i} (1-5)\t: "))
62                 items.append((name, weight, priority))
63             canBeSplit = input(
64                 "\nDo you want to break an item into several pieces? (Yes/No): ").lower() == 'yes'
65
66             if canBeSplit:
67                 fractionalKnapsack(selectedTrainClass, items)
68             else:
69                 oneZeroKnapsack(selectedTrainClass, items)
70
71         if input("\nWant to try another class? (Yes/No): ").lower() != 'yes':
72             break

```

4. 0/1 Knapsack

```
1 def oneZeroKnapsack(selectedTrainClass, items):
2     totalWeight = 0
3     totalValue = 0
4     itemRecommendations = []
5
6     for name, weight, priority in items:
7         if selectedTrainClass['trainCapacity'] <= 0:
8             break
9
10        if weight <= selectedTrainClass['trainCapacity']:
11            totalWeight += weight
12            totalValue += priority
13
14            itemRecommendations.insert(0, (name))
15            selectedTrainClass['trainCapacity'] -= weight
16
17    print("\nWe recommend you to bring :")
18    for idx, item in enumerate(itemRecommendations, start=1):
19        print(f"{idx}. {item}")
20    print(f"\nTotal weight of items carried: {totalWeight} kg")
```

5. Fractional Knapsack

```

1 def fractionalKnapsack(selectedTrainClass, items):
2     items.sort(key=lambda x: x[1], reverse=True)
3
4     totalWeight = 0
5     totalValue = 0
6     itemRecommendations = []
7
8     for name, weight, priority in items:
9         if selectedTrainClass['trainCapacity'] > 0:
10             if weight <= selectedTrainClass['trainCapacity']:
11                 selectedTrainClass['trainCapacity'] -= weight
12                 totalWeight += weight
13                 totalValue += priority
14                 itemRecommendations.append((name, "whole"))
15             else:
16                 bestItem = None
17                 sortItems = sorted(
18                     items, key=lambda x: x[1])
19                 for item in sortItems:
20                     if item[1] <= selectedTrainClass['trainCapacity']:
21                         fraction = min(
22                             1, selectedTrainClass['trainCapacity'] / item[1])
23                         weightTaken = fraction * item[1]
24                         totalWeight += weightTaken
25                         totalValue += item[2] * fraction
26                         if fraction == 1:
27                             fractionLabel = "whole"
28                         else:
29                             fractionLabel = str(
30                                 Fraction(fraction).limit_denominator())
31
32                         itemRecommendations.append(
33                             (item[0], fractionLabel))
34                         selectedTrainClass['trainCapacity'] -= weightTaken
35
36                         remainWeight = item[1] - weightTaken
37                         if remainWeight > 0:
38                             items.remove(item)
39                             items.append(
40                                 (item[0], remainWeight, item[2]))
41
42                         break
43
44                 if bestItem:
45                     selectedTrainClass['trainCapacity'] -= bestItem[1]
46                     totalWeight += bestItem[1]
47                     totalValue += bestItem[2]
48                     itemRecommendations.append(
49                         (bestItem[0], "whole"))
50                     items.remove(bestItem)
51                 else:
52                     remainCapacity = selectedTrainClass['trainCapacity']
53                     fraction = remainCapacity / weight
54                     weightTaken = fraction * weight
55                     totalWeight += weightTaken
56                     totalValue += priority * fraction
57                     itemRecommendations.append(
58                         (name, str(Fraction(round(fraction, 2)).limit_denominator()))
59                     selectedTrainClass['trainCapacity'] -= weightTaken
60
61                 for item in items:
62                     if item[1] <= selectedTrainClass['trainCapacity']:
63                         items.remove(item)
64                         items.insert(0, item)
65                         break
66
67             else:
68                 break
69
70     print("\nWe recommend items you can bring :")
71
72     for idx, item in enumerate(itemRecommendations, start=1):
73         nameItem = item[0].capitalize()
74         labelFr = item[1]
75
76         if labelFr != "whole":
77             print(f"{idx}. {nameItem} (whole)")
78         else:
79             print(f"{idx}. {nameItem} ({labelFr})")
80     print(f"\nTotal weight of items carried: {totalWeight} kg")

```

6. Pencetakan Tiket

Note : Untuk format tabelnya bukan tidak rapi ya miss, hanya saja kami membuatnya manual, tidak menggunakan fungsi tabulate. Jadi menyesuaikan dengan outputnya.

```
1 def orderTicket():
2     print("\nMake sure you have checked the route before booking tickets.\nWhere are you going ?")
3     originStation = input("From : ")
4     destinationStation = input("To : ")
5     if originStation not in stations or destinationStation not in stations:
6         print("Invalid station.")
7         return
8
9     routeDuration, track = dijkstra(route, originStation, destinationStation)
10    if routeDuration == float("inf"):
11        print("No routes available.")
12        return
13
14    print(
15        f"\nThis is the travel route from {originStation} to {destinationStation}: \n{' -> '.join(track)} \nDuration: {routeDuration} hours")
16
17    while True:
18        print("\nMake sure you have checked your luggage before choosing a class.")
19        trainClassInput = input(
20            "Select class (Economy, Business, Exclusive): ")
21        if trainClassInput.lower() in ['economy', 'business', 'exclusive']:
22            trainClass = trainClassInput.capitalize()
23            break
24        else:
25            print("The train class you entered is incorrect. Please try again.")
26
27        print("\nPlease enter your data : ")
28        name = input("Enter your name\t\t\t\t: ")
29        date = input("Enter the departure date (DD/MM/YYYY)\t: ")
30        time = input("Enter departure time (HH:MM)\t\t: ")
31        confirmation = input("Is your data correct? (Yes/No)\t\t: ")
32
33        trainNumber = random.randint(000, 999)
34        trainn = random.choice(train)
35        platform = f"{random.randint(1, 10):02}"
36        seat = random.randint(1, 50)
37        timeArrive = datetime.strptime(
38            date + ' ' + time, '%d/%m/%Y %H:%M') + timedelta(hours=routeDuration)
39        travelTime = datetime.strptime(
40            date + ' ' + time, '%d/%m/%Y %H:%M')
41
42        tp = travelTime.strftime(
43            '%d/%m/%Y')
44        wp = travelTime.strftime('%H:%M')
45        tt = timeArrive.strftime('%d/%m/%Y')
46        wt = timeArrive.strftime('%H:%M')
47        totalCost = 15 * routeDuration + trainn['trainClassPrices']
48
49        if confirmation.lower() == 'yes':
50            print(f'''
51            +-----+
52            | MELLOW TRAIN TICKETS |
53            +-----+
54            | ORIGIN      : {originStation} |
55            +-----+
56            | DATE       : {tp}          TIME : {wp} |
57            | TRAIN#    : {trainNumber}          CLASS :{trainClass} |
58            | PLATFORM  : {platform}          SEAT : {seat} |
59            +-----+
60            | DESTINATION : {destinationStation} |
61            +-----+
62            | DATE       : {tt}          TIME : {wt} |
63            +-----+
64            | PASSENGER NAME : {name} |
65            +-----+
66            ''')
67            print(f"Total cost : ${totalCost}")
68            if input("\nOrdering more tickets? (Yes/No): ").lower() != "Yes":
69                pass
70            else:
71                orderTicket()
72        else:
73            print("\nTicket order cancelled. Please try again.")
```

Algoritma dan Kompleksitas

1. Algoritma

```
1 # ALGORITMA
2
3 # 1. IMPORT HEAPQ, RANDOM, DATETIME, TIMEDELTA, TABULATE, FRACTION.
4 # 2. BUAT LIST station, DICT SELURUH route DAN DICT train DIDALAM LIST.
```

```
1 # 3. INISIALISASI DEF DIJKSTRA DENGAN 3 PARAMETER (route, begin, end).
2 # - INISIALISASI queue DENGAN LIST YANG MENGISI TIAP PARAMETER DIJKSTRA.
3 # - SET visited DENGAN FUNGSI SET().
4 # - LAKUKAN PERULANGAN queue SELAMA KONDISI TERPENUHI.
5 # - INISIALISASI routeDuration, station, JALUR KE DALAM VARIABEL queue YANG TELAH DIBUAT SEBELUMNYA.
6 # - JIKA station TERMASUK DALAM visited, TERUSKAN PROGRAM.
7 # - TAMBAHKAN station KE DALAM visited.
8 # - TAMBAHKAN JALUR DENGAN station UNTUK DIMASUKKAN NILAINYA KE DALAM JALUR.
9 # - JIKA station == end, KEMBALIKAN NILAI routeDuration DAN JALUR.
10 # - SELAMA nextStation DAN distance BERADA DIDALAM route, AMBIL NILAI station PER-ITEM.
11 # - JIKA nextStation TIDAK ADA DI visited,
12 # - TAMBAHKAN NILAI YANG ADA DIDALAM routeDuration+distance, nextStation, JALUR KE DALAM queue.
13 # - TAMBAHKAN KE BAGIAN BELAKANG.
14 # - KEMBALIKAN NILAINYA.
```

```
1 # 4. INISIALISASI DEF VIEWTRAIN.
2 # - LAKUKAN PERULANGAN SELAMA KONDISI BENAR.
3 # - INISIALISASI table.
4 # - LAKUKAN PERULANGAN train SELAMA NILAI YANG ADA DIDALAM VARIABLE train.
5 # - BUAT table MENGGUNAKAN TABULATE.
6 # - JIKA PENGGUNA TIDAK INGIN DIBERIKAN goodsCarried ALIAS MENGINPUT "TIDAK",
7 # - MAKA BERIKAN STEP BERIKUTNYA: TANYA trainClassName DAN KEMUDIAN KELUAR DARI MENU VIEWTRAIN.
8 # - JIKA PENGGUNA MENGINPUT DATA YANG SALAH (DILUAR DARI KATA YA/TIDAK),
9 # - BERIKAN KESEMPATAN UNTUK MENGULANG YANG BENAR.
10 # - JIKA PENGGUNA INGIN DIBERIKAN goodsCarried ALIAS MENGINPUT "YA",
11 # - MAKA BERIKAN STEP BERIKUTNYA YAITU TANYAKAN trainClassName API YANG INGIN DICoba.
12 # - CETAK PENJELASAN UNTUK MEMBERIKAN NILAI PRIORITAS MASING-MASING BARANG.
13 # - TANYAKAN BERAPA BANYAK BARANG YANG INGIN DIBAWA.
14 # - LAKUKAN PERULANGAN I SEPANJANG amountOfGoods + 1.
15 # - CETAK FORMAT UNTUK MEMBUAT DATA BARANG KEMUDIAN MASUKKAN KE items.
16 # - CETAK DAN INPUT APAKAH BARANG INGIN DIPECAH?.
17 # - JIKA YA, MAKA LANJUTKAN FUNGSI FRACTIONALKNAPSACK.
18 # - JIKA TIDAK, MAKA LANJUTKAN FUNGSI ONEZEROKNAPSACK.
19 # - TANYAKAN APAKAH INGIN MENCOBA KELAS LAIN?
20 # - JIKA YA, MAKA ULANGI DEF VIEWTRAIN, JIKA TIDAK MAKA PROGRAM BERHENTI.
```

```

1 # 5. INISIALISASI DEF ONEZEROKNAPSACK DENGAN 2 PARAMETER (selectedTrainClass,items)
2 # - INISIALISASI NILAI totalWeight = 0
3 # - INISIALISASI NILAI totalValue = 0
4 # - INISIALISASI NILAI itemRecommendation DENGAN LIST KOSONG
5 # - LAKUKAN PERULANGAN NAMA,BERAT,PRIORITAS SELAMA NILAINYA ADA DI VARIABEL items.
6 # - JIKA selectedTrainClass LEBIH KECIL SAMA DENGAN 0, HENTIKAN.
7 # - JIKA BERAT <= selectedTrainClass,
8 # - TAMBAHKAN BERAT KE VARIABEL totalWeight DAN TAMBAHKAN NILAI PRIORITAS KE DALAM VARIABEL totalValue.
9 # - TAMBAHKAN NAMA KE DALAM itemRecommendation.
10 # - KURANGKAN BERAT DENGAN NILAI selectedTrainClass KEMUDIAN MASUKKAN NILAINYA DI selectedTrainClass.
11 # - CETAK KAMI MEREKOMENDASIKAN KAMU UNTUK MEMBAWA :.
12 # - LAKUKAN PERULANGAN UNTUK MENCETAK NOMOR DAN NAMA BARANG.
13 # - CETAK BERAT TOTAL BARANG YANG DIBAWA DARI NILAI YANG ADA DI VARIABEL TOTAL BERAT.

```

```

1 # 6. INISIALISASI DEF FRACTIONALKNAPSACK DENGAN 2 PARAMETER (selectedTrainClass, items).
2 # - MENGURUTKAN items.
3 # - INISIALISASI VARIABEL totalWeight DAN totalValue DENGAN NILAI 0.
4 # - INISIALISASI VARIABEL itemRecommendation DENGAN LIST KOSONG.
5 # - LAKUKAN PERULANGAN NAMA,BERAT,PRIORITAS SELAMA NILAINYA ADA DI items.
6 # - JIKA BERAT <= selectedTrainClass,
7 # - KURANGKAN BERATNYA DAN TAMBAHKAN BERAT KE TOTAL BERAT SERTA PRIORITAS KE TOTAL NILAI.
8 # - TAMBAHKAN NILAI YANG ADA DI VARIABEL NAMA DIKUTI DENGAN KATA whole DI itemRecommendation.
9 # - JIKA TIDAK, INISIALISASI NILAI bestItem DENGAN NONE DAN URUTKAN items.
10 # - SELAMA ITEM ADA DI sortItems, JIKA ITEM PADA INDEKS PERTAMA LEBIH KECIL DARI selectedTrainClass, HITUNG fractionNYA.
11 # - MASUKKAN NILAI weightTaken KE DALAM totalWeight.
12 # - MASUKKAN HASIL PERKALIAN DARI ITEM PADA INDEKS KEDUA DENGAN fraction KE DALAM totalValue.
13 # - JIKA fraction SAMA DENGAN 1, MAKA CETAK whole. JIKA TIDAK MAKA CETAK NILAI PECAHAN MENGGUNAKAN LIBRARY FRACTION.
14 # - TAMBAHKAN NILAINYA KEDALAM VARIABEL itemRecommendation.
15 # - KURANGKAN NILAI ITEM DI INDEKS PERTAMA DENGAN BERAT YANG DIAMBIL DAN MASUKKAN KEDALAM remainWeight.
16 # - JIKA remainWeight > 0, HAPUS ITEM items DAN ITEM INDEKS-0, remainWeight DAN ITEM PADA INDEKS-2, KE remainWeight.
17 # - HENTIKAN PROGRAM.
18 # - JIKA bestItem, LAKUKAN PENGURANGAN INDEKS PERTAMA-1 PADA bestItem DAN TAMBAHKAN KEDALAM selectedTrainClass.
19 # - TAMBAHKAN INDEKS-1 bestItem KE totalWeight DAN TAMBAH INDEKS-2 bestItem KE totalValue.
20 # - MASUKKAN INDEKS KE NOL DENGAN KATA whole KEDALAM itemRecommendation.
21 # - JIKA TIDAK, MASUKKAN NILAI DARI selectedTrainClass KEDALAM SISAKAPASITAS.
22 # - BAGI NILAI SISAKAPASITAS DENGAN BERAT DAN MASUKKAN NILAINYA KEDALAM VARIABEL fraction.
23 # - KALIKAN NILAI fraction DENGAN BERAT DAN TAMPUNG NILAINYA KE DALAM weightTaken.
24 # - TAMBAHKAN NILAI weightTaken KE DALAM VARIABEL totalWeight.
25 # - TAMBAHKAN HASIL PERKALIAN fraction DENGAN PRIORITAS KE DALAM VARIABEL totalValue.
26 # - MASUKKAN NILAI NAMA DAN fraction KE DALAM itemRecommendation.
27 # - KURANGKAN NILAI weightTaken DENGAN selectedTrainClass.
28 # - LAKUKAN PERULANGAN ITEM SELAMA NILAINYA ADA PADA items.
29 # - JIKA ITEM PADA INDEKS PERTAMA LEBIH KECIL SAMA DENGAN selectedTrainClass,
30 # - HAPUS ITEM PADA items DAN MASUKKAN KEMBALI PADA POSISI PERTAMA.
31 # - JIKA selectedTrainClass BUKAN LEBIH BESAR DARI NOL, HENTIKAN PROGRAM TERSEBUT.
32 # - CETAK KAMI MEREKOMENDASIKAN BARANG YANG DAPAT DIBAWA.
33 # - LAKUKAN PERULANGAN UNTUK MENCETAK HASILNYA.
34 # - CETAK BERAT TOTAL BARANG YANG DIBAWA.

```



```
1 # 7. INISIALISASI FUNGSI VIEWROUTES.
2 # - INISIASASI stations DENGAN LIST UNTUK route ALIAS AMBIL NILAI KEYNYA.
3 # - INISIALISASI VARIABEL TELAHDITAMPILKAN DENGAN NILAI SET.
4 # - INISIALISASI VARIABEL UNTUK table.
5 # - LAKUKAN PERULANGAN SELAMA begin ADA NILAINYA DI stationS.
6 # - LAKUKAN PERULANGAN end DARI INDEKS begin DI route.
7 # - JIKA VAR begin DAN end TIDAK ADA DI TELAHDITAMPILKAN, BEGITU JUGA SEBALIKNYA,
8 # - INISIALISASI routeDuration UNTUK MENAMPUNG NILAI route DI INDEKS begin DAN end.
9 # - INISIALISASI table DENGAN FUNGSI TABULATE UNTUK MENCETAK route DAN duration.
10 # - LAKUKAN PERULANGAN SELAMA KONDISI BENAR.
11 # - CETAK MASUKKAN route KAMU.
12 # - INPUT NILAI originStation DAN destination.
13 # - JIKA originStation TIDAK ADA DI DAFTAR stationS, CETAK station availed. SILAKAN COBA LAGI.
14 # - PANGGIL FUNGSI DJIKSTRA DAN TAMPUNG NILAINYA DI VARIABEL routeDuration DAN JALUR.
15 # - JIKA routeDuration NILAINYA TAK HINGGA, CETAK TIDAK ADA route YANG TERSEDIA.
16 # - JIKA TIDAK, PRINT JALUR TERPENDEKNYA.
17 # - JIKA INPUT LIHAT route LAIN BUKAN 'YA', HENTIKAN PROGRAM TERSEBUT.
```



```
1 # 8. INISIALISASI DEF ORDERTICKET.
2 # - CETAK PASTIKAN ANDA TELAH MEMERIKSA route SEBELUM MEMESAN TIKET DAN TANYAKAN KEMANA AKAN PERGI.
3 # - INPUT NILAI originStation DAN destination.
4 # - JIKA station ASAL TIDAK ADA DI DAFTAR station, CETAK station availed DAN KEMBALIKAN NILAINYA.
5 # - PANGGIL FUNGSI DJIKSTRA DAN TAMPUNG KE DALAM VARIABEL routeDuration DAN track.
6 # - JIKA NILAI routeDuration INF, CETAK TIDAK ADA route YANG TERSEDIA DAN KEMBALIKAN NILAI.
7 # - CETAK route PERJALANAN TERPENDEK.
8 # - LAKUKAN PERULANGAN SELAMA KONDISI BENAR.
9 # - CETAK PASTIKAN ANDA TELAH MEMERIKSA BARANGBAWAAN ANDA SEBELUM MEMILIH KELAS.
10 # - INPUT NILAI trainClassAPI.
11 # - JIKA TULISANNYA LOWER SEMUA, MAKA CAPITALIZE DAN BREAK.
12 # - JIKA TIDAK, CETAK trainClassName YANG ANDA MASUKKAN SALAH. SILAKAN COBA LAGI.
13 # - CETAK SILAKAN MASUKKAN DATA ANDA.
14 # - MASUKKAN DATA name, date, time DAN confirmation.
15 # - INISIALISASI numberTrain, train, platform, seat DENGAN NILAI ACAK MENGGUNAKAN FUNGSI RANDOM.
16 # - INISIALISASI FORMAT timeArrive DAN travelTime.
17 # - INISIALISASI travelTime DENGAN TP DAN WP.
18 # - INISIALISASI timeArrive DENGAN TT DAN WT.
19 # - HITUNG totalCost DENGAN MENGALIKAN 15 DENGAN routeDuration + train.
20 # - JIKA KONFIRMASI SAMA DENGAN 'YA', CETAK FORMAT TIKET.
21 # - CETAK TOTAL BIAYA MELALUI VARIABEL totalCost.
22 # - JIKA PESAN TIKET LAGI DIINPUT YA, MAKA JALANKAN KEMBALI DEF ORDERTICKET.
23 # - JIKA BUKAN, MAKA BERHENTI DAN CETAK PEMESANAN TIKET DIBATALKAN. SILAKAN COBA LAGI.
```

```

1 # 9. INISIALISASI DEF mainMenu.
2 # - CETAK SELAMAT DATANG DI PROGRAM PEMESANAN TIKET KERETA API.
3 # - INPUT NAMA DEPAN.
4 # - LAKUKAN PERULANGAN SELAMA KONDISI BENAR.
5 # - CETAK SELAMAT DATANG DI (NAMA KITA) EXPRESS.
6 # - CETAK SEMUA MENU (PESAN TIKET, LIHAT route, LIHAT train, KELUAR).
7 # - INPUT UNTUK MEMILIH choice MENU.
8 # - JIKA MEMILIH 1, JALANKAN FUNGSI ORDERTICKET.
9 # - JIKA MEMILIH 2, JALANKAN FUNGSI VIEWROUTES.
10 # - JIKA MEMILIH 3, JALANKAN FUNGSI VIEWTRAIN.
11 # - JIKA MEMILIH 4, CETAK TERIMA KASIH TELAH MENGGUNAKAN PROGRAM KAMI. HENTIKAN PROGRAM.
12 # - JIKA BUKAN SEMUANYA, CETAK choice aailed. SILAKAN COBA LAGI.
13 # 10. JALANKAN FUNGSI mainMenu.

```

2. Kompleksitas

```

1 # FUNGSI MAINMENU
2 #  $T(n)$  = Tergantung fungsi yang dipanggil dalam loop
3 # Fungsi ini memiliki loop utama yang terus berjalan selama pengguna tidak memilih opsi Exit (opsi 4).
4 # Setiap pemilihan opsi akan memanggil fungsi yang sesuai,
5 # sehingga kompleksitas waktunya bergantung pada fungsi dari opsi yang dipilih.
6 #  $S(n)$  = Tergantung fungsi yang dipanggil
7 # Variabel name yang memakan ruang konstan  $O(1)$ .
8 # Loop utama yang tidak memiliki struktur data tambahan sehingga ruang yang digunakan adalah  $O(1)$ .
9 # Pemanggilan fungsi dari opsi yang dipilih,
10 # dari pemilihan opsi itulah kompleksitas ruangnya dihitung (sesuai fungsi yang dipanggil).

```

```

1 # FUNGSI DEF FRACTIONAL KNAPSACK
2 #  $T(n) = O(n \log n)$ 
3 # Operasi pengurutan barang menggunakan sort dengan kompleksitas  $O(n \log n)$ .
4 # Loop for untuk memproses barang yang mengunjungi tiap item dalam items yang memiliki panjang n yang berarti Loop berjalan
5 # sebanyak n kali. Namun, di Loop ini ada operasi sorting lagi yang memakan  $O(n \log n)$  dalam kasus terburuknya.
6 # Oleh karena itu, keseluruhan kompleksitas waktu dalam fungsi ini didominasi dan dihasilkan oleh operasi pengurutannya.
7 #  $S(n) = O(n)$ 
8 # Variabel dari (totalWeight dan totalValue) yang memakan ruang  $O(1)$ .
9 # List itemRecommendations yang dapat menyimpan hingga n item jika semua barang direkomendasikan untuk dibawa, hingga memakan ruang  $O(n)$ .
10 # List items yang menyimpan daftar barang hingga memakan ruang  $O(n)$ .
11 # Hasil kompleksitas ruang dari fungsi ini didapat dari gabungan ruang yang digunakan oleh List itemRecommendations dan List items.

```

```

1  # FUNGSI DEF ONEZEROKNAPSACK
2  #  $T(n) = O(n)$ 
3  # Loop for yang mengunjungi setiap item dalam items yang memiliki panjang n, yang berarti Loop berjalan sebanyak n kali.
4  # Setiap operasi dalam Loop merupakan operasi konstan yang bernilai  $O(1)$ .
5  # Hasil kompleksitas waktu dari fungsi ini merupakan hasil dari Loop yang mengunjungi tadi.
6  #  $S(n) = O(n)$ 
7  # Variabel dari (totalWeight dan totalValue) yang memakan ruang  $O(1)$ .
8  # List itemRecommendations yang dapat menyimpan hingga n item jika semua barang direkomendasikan untuk dibawa,
9  # hingga memakan ruang  $O(n)$ .
10 # Hasil kompleksitas ruang dari fungsi ini didapat dari ruang yang digunakan oleh List itemRecommendations.

```

```

1  # FUNGSI DEF VIEWTRAIN
2  #  $T(n) = O(n)$ 
3  # Membuat tabel loop for train untuk mengunjungi setiap item dalam List train yang memiliki panjang n =>  $O(n)$ .
4  # Operasi meminta inputan pengguna yang merupakan operasi konstan yang tidak bergantung pada ukuran input =>  $O(1)$ .
5  # Pemilihan kelas kereta yang menggunakan next dengan kondisi dalam List train yang panjangnya n =>  $O(n)$ .
6  # Pengolahan barang yang membaca situasi jika barang dibawa akan ada Loop yang meminta pengguna memasukkan informasi barang
7  # yang panjangnya juga n =>  $O(n)$ .
8  # Hasil kompleksitas waktu merupakan hasil dari kasus terburuknya, yang dimana semua operasi akan dijalankan dalam urutan n.
9  #  $S(n) = O(n)$ 
10 # List table menyimpan informasi tentang setiap kereta dalam train, sehingga ruang =>  $O(n)$ .
11 # Variabel selectedTrainClass menyimpan satu elemen dari List train, sehingga ruang =>  $O(1)$ .
12 # List items menyimpan informasi tentang setiap barang yang dimasukkan pengguna, yang bisa memiliki panjang n,
13 # sehingga diperlukan ruang sebesar  $O(n)$ .
14 # Total ruang untuk fungsi ini diambil dari ruang yang digunakan oleh List table dan items.

```

```

1  # FUNGSI DEF DIJKSTRA
2  #  $T(n) = O((V + E) \log V)$ 
3  # Menghapus dan menambahkan elemen ke priority queue membutuhkan  $O(\log V)$ .
4  # Ada V node yang akan dimasukkan dan dihapus dari priority queue, sehingga dibutuhkan  $O(V \log V)$ .
5  # Untuk mengunjungi setiap edge (jumlah E), periksa tetangga dan mungkin menambahkannya ke queue
6  # yang juga membutuhkan  $O(\log V)$ , sehingga totalnya menjadi  $O(E \log V)$ .
7  # Gabungan dari faktor ini menghasilkan kompleksitas waktu fungsi ini.
8  #  $S(n) = O(V)$ 
9  # Bisa menyimpan hingga V elemen pada priority queue hingga membutuhkan ruang  $O(V)$ .
10 # Bisa menyimpan hingga V elemen pada set visited hingga membutuhkan ruang  $O(V)$ .
11 # Maksimal pada track List berisi V node hingga membutuhkan ruang  $O(V)$ .
12 # Gabungan dari ketiganya merupakan total ruang yang digunakan untuk fungsi ini.

```



```

1 # FUNGSI DEF VIEWROUTES
2 #  $T(n) = O(V + E)$ 
3 # Tabel rute yang mengandung looping stasiun dimana loop pertama mengunjungi tiap stasiun dalam route yang memiliki panjang V.
4 # Loop kedua yang mengunjungi tiap rute dari stasiun tersebut yang totalnya ada E, sehingga kompleksitasnya menggunakan  $O(V + E)$ .
5 # Algo dijkstra yang dijalankan untuk mencari rute terpendek anatar dua stasiun dengan kompleksitas  $O((V + E)\log V)$ , namun dalam
6 # viewRoutes dijalankan beberapa kali tergantung input pengguna, tetapi tiap kali input pengguna tidak mempengaruhi V dan E.
7 # Jadi, kompleksitasnya adalah  $O(V + E)$ .
8 # Kompleksitas waktu untuk fungsi ini didapat dari gabungan antara tabel rute yang mengandung Looping stasiun dan algo dijkstra.
9 #  $S(n) = O(V + E)$ 
10 # List stations yang menyimpan semua stasiun dalam route sehingga membutuhkan ruang  $O(V)$ .
11 # Set displayed yang meyimpan pasangan rute yang sudah ditampilkan, maksimal menyimpan E pasangan, sehingga membutuhkan ruang  $O(E)$ .
12 # List table yang menyimpan informasi rute yang ditampilkan, maksimal menyimpan E rute, sehingga membutuhkan ruang  $O(E)$ .
13 # Hasil kompleksitas ruang fungsi ini diperoleh dari gabungan ketiga ruang yang digunakan oleh ketiga elemen di atas.

```

```

1 # FUNGSI ORDERTICKET
2 #  $T(n) = O(1)$ 
3 # Validasi stasiun yang mengecek apakah stasiun asal dan tujuan valid yang merupakan operasi konstan memakan waktu  $O(1)$ .
4 # Algo dijkstra yang dianggap sebagai bagian dari input tetap karena route adalah graf yang tidak berubah selama pemesanan
5 # tiket sehingga tidak ada pengaruhnya dalam kompleksitas waktu fungsi ini.
6 # Mengambil input dari pengguna dan memprosesnya yang merupakan operasi konstan sehingga memakan waktu  $O(1)$ .
7 # Operasi untuk menentukan nomor kereta, platform, dan nomor tempat duduk dengan nilai acak, yang operasinya juga konstan
8 # sehingga memakan waktu  $O(1)$ .
9 # Hasil kompleksitas waktu fungsi ini didapat dari penggabungan ketiga elemen yang memiliki kompleksitas waktu di atas.
10 #  $S(n) = O(V + E)$ 
11 # Data stations yang digunakan untuk validasi bisa menyimpan hingga V elemen.
12 # Algo dijkstra yang dijalankan dalm konteks fungsi ini membutuhkan ruang  $O(V + E)$  untuk menyimpan jalur dan durasi rute.
13 # Variabel Lokal yang menyimpan informasi input pengguna, hasil pemesanan, dan informasi acak lainnya membutuhkan ruang konstan  $O(1)$ .
14 # Total kompleksitas ruang fungsi ini diambil dari algo dijkstranya karena tidak ada ruang tambahan yang signifikan di Luar itu.

```

```

1 # TOTAL KOMPLEKSITAS WAKTU
2 #  $T(n) = O((V + E)\log V + n \log n)$ 
3 # CATATAN :
4 # PADA SETIAP FUNGSI, AKAN DIPILIH WAKTU YANG PALING EFEKTIF DAN EFISIEN.
5 # FUNGSI YANG PALING BAGUS ADA DI ALGORITMA DIJKSTRA DAN FRACTIONAL KNAPSACK DI DALAM MENU orderTicket(), viewRoutes() DAN 'viewTrain()'.
6 # SEHINGGA, KOMPLEKSITAS WAKTU SECARA KESELURUHAN YANG DIDAPAT MERUPAKAN PENGABUNGAN ANTARA ALGORITMA DIJKSTRA DAN FRACTIONAL KNAPSACK.
7
8
9 # TOTAL KOMPLEKSITAS RUANG
10 #  $S(n) = O(V + E + n)$ 
11 # PADA SETIAP FUNGSI, AKAN DIPILIH RUANG YANG PALING BESAR.
12 # FUNGSI DENGAN PENGGUNAAN RUANG TERBESAR ADA PADA dijkstra(), viewRoutes(), oneZeroKnapsack() DAN fractionalKnapsack().
13 # SEHINGGA, KOMPLEKSITAS RUANG SECARA KESELURUHAN YANG DIDAPAT MERUPAKAN PENGABUNGAN ANTARA KEEMPAT FUNGSI TERSEBUT.

```

Hasil Pengujian

1. Menu Utama

```
=====
Welcome to the Train ticket booking program

Before using the program, enter your first name: Mellow
=====

Welcome to Mellow Express
=====
1. Order Ticket
2. View Routes
3. View Train
4. Exit
=====
choose (1-4): █
```

2. Menu Penentuan Rute

```
Cost : $15/jam
+-----+-----+
| Routes (2-way) | Duration (Hours) |
+-----+-----+
| Minstowe -- Cowstone | 3 |
| Oldcastle -- New North | 5 |
| Oldcastle -- Freeham | 2 |
| Cowstone -- New North | 4 |
| Cowstone -- Bingborough | 6 |
| Cowstone -- Donningpool | 7 |
| Cowstone -- Highbrook | 5 |
| Cowstone -- Freeham | 2 |
| New North -- Bingborough | 3 |
| New North -- Donningpool | 6 |
| New North -- Wington | 4 |
| New North -- Highbrook | 2 |
| Freeham -- Donningpool | 3 |
| Freeham -- Wington | 5 |
| Bingborough -- Donningpool | 2 |
| Bingborough -- Highbrook | 1 |
| Donningpool -- Wington | 4 |
| Donningpool -- Highbrook | 5 |
| Donningpool -- Old Mere | 2 |
+-----+-----+
Enter your route
=====

From : Minstowe
To : Old Mere

This is the shortest route from Minstowe to Old Mere:
Minstowe -> Cowstone -> Freeham -> Donningpool -> Old Mere
Duration: 10 hours

See another route? (Yes/No): No

Welcome to Mellow Express
=====
1. Order Ticket
2. View Routes
3. View Train
4. Exit
=====
choose (1-4): █
```


Cost : \$15/jam

Routes (2-way)	Duration (Hours)
Minstowe -- Cowstone	3
Oldcastle -- New North	5
Oldcastle -- Freeham	2
Cowstone -- New North	4
Cowstone -- Bingborough	6
Cowstone -- Donningpool	7
Cowstone -- Highbrook	5
Cowstone -- Freeham	2
New North -- Bingborough	3
New North -- Donningpool	6
New North -- Wington	4
New North -- Highbrook	2
Freeham -- Donningpool	3
Freeham -- Wington	5
Bingborough -- Donningpool	2
Bingborough -- Highbrook	1
Donningpool -- Wington	4
Donningpool -- Highbrook	5
Donningpool -- Old Mere	2

Enter your route

=====

From : Minstowe

To : Old Mere

This is the shortest route from Minstowe to Old Mere:

Minstowe -> Cowstone -> Freeham -> Donningpool -> Old Mere

Duration: 10 hours

See another route? (Yes/No): Yes

Enter your route

=====

From : Freeham

To : Cowstone

This is the shortest route from Freeham to Cowstone:

Freeham -> Cowstone

Duration: 2 hours

3. Menu Penentuan Kereta Api

```
+-----+-----+-----+
| Train Name | Capacity (kg) | Price ($) per kg |
+-----+-----+-----+
| Economy    | 25            | 20                |
+-----+-----+-----+
| Business   | 30            | 30                |
+-----+-----+-----+
| Exclusive  | 35            | 40                |
+-----+-----+-----+

we can help you choose what to bring, whether you want to try ?
yes/no = no
Select Class : Business

Welcome to Mellow Express
=====
1. Order Ticket
2. View Routes
3. View Train
4. Exit
=====
choose (1-4): █
```

```
+-----+-----+-----+
| Train Name | Capacity (kg) | Price ($) per kg |
+-----+-----+-----+
| Economy    | 25            | 20                |
+-----+-----+-----+
| Business   | 30            | 30                |
+-----+-----+-----+
| Exclusive  | 35            | 40                |
+-----+-----+-----+

we can help you choose what to bring, whether you want to try ?
yes/no = niovf
you entered it wrong. try again

+-----+-----+-----+
| Train Name | Capacity (kg) | Price ($) per kg |
+-----+-----+-----+
| Economy    | 25            | 20                |
+-----+-----+-----+
| Business   | 30            | 30                |
+-----+-----+-----+
| Exclusive  | 35            | 40                |
+-----+-----+-----+

we can help you choose what to bring, whether you want to try ?
yes/no = yes
Select Class : Economy
Give your items a priority scale ranging from 1 (very important) to 5 (not important)
How many items do you want to bring? 5
```

Item-1
Name of Item-1 : Book
Weight of Item-1 (kg) : 10
Priority Of Item-1 (1-5) : 1

Item-2
Name of Item-2 : Clothes
Weight of Item-2 (kg) : 12
Priority Of Item-2 (1-5) : 1

Item-3
Name of Item-3 : Snack
Weight of Item-3 (kg) : 4
Priority Of Item-3 (1-5) : 4

Item-4
Name of Item-4 : Souvenir
Weight of Item-4 (kg) : 3
Priority Of Item-4 (1-5) : 2

Item-5
Name of Item-5 : Cosmetics
Weight of Item-5 (kg) : 1
Priority Of Item-5 (1-5) : 3

Do you want to break an item into several pieces? (Yes/No): No

We recommend you to bring :

1. Souvenir
2. Clothes
3. Book

Total weight of items carried: 25.0 kg

Item-1

Name of Item-1 : Book

Weight of Item-1 (kg) : 10

Priority Of Item-1 (1-5) : 1

Item-2

Name of Item-2 : Clothes

Weight of Item-2 (kg) : 12

Priority Of Item-2 (1-5) : 1

Item-3

Name of Item-3 : Snack

Weight of Item-3 (kg) : 4

Priority Of Item-3 (1-5) : 4

Item-4

Name of Item-4 : Souvenir

Weight of Item-4 (kg) : 3

Priority Of Item-4 (1-5) : 2

Item-5

Name of Item-5 : Cosmetics

Weight of Item-5 (kg) : 1

Priority Of Item-5 (1-5) : 3

Do you want to break an item into several pieces? (Yes/No): Yes

We recommend items you can bring :

1. Clothes (whole)
2. Book (whole)
3. Cosmetics (whole)
4. Snack (1/2)

Total weight of items carried: 25.0 kg

4. Menu Pencetakan Tiket

```
Make sure you have checked the route before booking tickets.
Where are you going ?
From : Minstowe
To   : Old Mere

This is the travel route from Minstowe to Old Mere:
Minstowe -> Cowstone -> Freeham -> Donningpool -> Old Mere
Duration: 10 hours

Make sure you have checked your luggage before choosing a class.
Select class (Economy, Business, Exclusive): Excclusive
The train class you entered is incorrect. Please try again.

Make sure you have checked your luggage before choosing a class.
Select class (Economy, Business, Exclusive): Exclusive

Please enter your data :
Enter your name           : Tasya
Enter the departure date (DD/MM/YYYY) : 23/06/2023
Enter departure time (HH:MM)         : 23:00
Is your data correct? (Yes/No)       : Yes

+-----+
| MELLOW TRAIN TICKETS |
+-----+
| ORIGIN      : Minstowe |
+-----+
| DATE        : 23/06/2023   TIME : 23:00 |
| TRAIN#      : 940          CLASS : Exclusive |
| PLATFORM    : 07          SEAT  : 11 |
+-----+
| DESTINATION : Old Mere |
+-----+
| DATE        : 24/06/2023   TIME : 09:00 |
+-----+
| PASSENGER NAME : Tasya |
+-----+

Total cost : $190
```

3. Menu Keluar dari Program

```
Welcome to Mellow Express
=====
1. Order Ticket
2. View Routes
3. View Train
4. Exit
=====
choose (1-4): 4
Thank you for using our program.
PS C:\Users\simba\OneDrive\KULIAH DIMARE\PEMIKIRAN KOMPUTASI\UAS>
```