Marks distribution for testing exercise 2			TOTAL
Hiba	Alicia & Álvaro	Lucía & Irem	
3.33	3.33	3.33	10

1) Write at least the pseudocode corresponding to the identified method or methods.

```
public static TypeFare getFare(AirlineCustomer c) {
	TypeFare fare = null;
	if (c.getAge() < 18 && c.getFlightsPerYear() >= 6) {
		fare = new TypeFare("Pajarillo", 10);
	}

	if (isStudent(c)) {
		fare = new TypeFare("Gorrión", 15);
	}

	if (isYoungWorker(c)) {
		if (c.isLivesWithParents()) {
		fare = new TypeFare("Travel While You Can", 5);
		} else {
			fare = new TypeFare("Daring to Leave the Nest", 25);
		}
	}

	if (isMidIncomeAdult(c)) {
		if (c.isTravelWithChildren()) {
			fare = new TypeFare("Discover Europe with Your Little Ones", 10);
		} else {
			fare = new TypeFare("Discover Europe", 15);
		}
	}

	if (isHighIncomeAdult(c)) {
		if (c.isTravelWithChildren()) {
			fare = new TypeFare("Discover the World with Your Little Ones", 10);
		} else {
			fare = new TypeFare("Discover the World", 20);
		}
	}
	return fare;
}

public static boolean isStudent(AirlineCustomer c) {
	boolean result = false;
	if(c.getAge() >= 18 && c.getAge() <= 25
		&& c.getTravelerType() == TravelerType.STUDENT
		&& c.getTravelClass() == TravelClass.ECONOMY
		&& c.getFlightsPerYear() >= 9) {
			result = true;
	}
	return result;
}

public static boolean isYoungWorker(AirlineCustomer c) {
```

```
        boolean result = false;
        if(c.getAge() >= 18 && c.getAge() <= 25
                && c.getTravelerType() == TravelerType.WORKER
                && c.getTravelClass() == TravelClass.ECONOMY
                && c.getFlightsPerYear() >= 3) {
                        result = true;
        }
        return result;
}

public static boolean isMidIncomeAdult(AirlineCustomer c) {
        boolean result = false;
        if(c.getAge() > 25
                && c.getIncome() >= 20000 && c.getIncome() < 35000
                && c.getTravelClass() == TravelClass.ECONOMY
                && c.getFlightsPerYear() >= 6
                && c.getDestination() == Destination.EUROPE) {
                        result = true;
        }
        return result;
}

public static boolean isHighIncomeAdult(AirlineCustomer c) {
        boolean result = false;
        if(c.getAge() > 25
                && c.getIncome() >= 35000
                && c.getTravelClass() == TravelClass.BUSINESS
                && c.getFlightsPerYear() >= 6
                && (c.getDestination() == Destination.ASIA
                || c.getDestination() == Destination.AMERICA)) {
                        result = true;
        }
        return result;
}
```

2) Identify the test units.

Every method is considered a test unit therefore we have in each of the clases:

FareRecommendationSystem:

- boolean isStudent(AirlineCustomer c)
- boolean isYoungWorker(AirlineCustomer c)
- boolean isMidIncomeAdult(AirlineCustomer c)
- boolean isHighIncomeAdult(AirlineCustomer c)
- TypeFare getFare(AirlineCustomer c)

3) Identify the variables that must be considered to test the relevant methods included in the test units.

Input Variables:

- age : int
- flightsPerYear : int

- travelerType : TravelerType (STUDENT, WORKER)
- travelClass : TravelClass (ECONOMY, BUSINESS)
- destination : Destination (EUROPE, ASIA, AMERICA, OTHER)
- income : double
- travelWithChildren : boolean
- livesWithParents : boolean

Output Variables:

- For the four helper methods: boolean (true / false)
- For getFare: TypeFare object, i.e. pair (name : String, discount : int) or null

4) Identify the test values for each of the above variables, which will constitute the corresponding test cases using the three techniques discussed in theory (equivalence classes, boundary value analysis, and error guessing), specifying for each value obtained which technique has been used.

FareRecommendationSystem

isStudent()

Variable	Partition	Limit	Guess	Final	Explanation
AirlineCustomer.getAge()	-8	-1,0,1	MININT -1 MAXINT +1 8.978 "nine" Null -8	11, 21, 30, 0, 18, 25	Some values are not included in the final set of values because they are being controlled in the command line interface.
	11	17,18,19			
	21	24,25,26			
	30				
AirlineCustomer.getTravelerType()	STUDENT WORKER		"neither" Null 7	STUDENT WORKER	
AirlineCustomer.getTravelClass()	ECONOMY BUSINESS		"other" 0 null	ECONOMY BUSINESS	
AirlineCustomer.getFlightsPerYear()	-10	-1,0,1	MININT –1 MAXINT +1 10.86 "eleven" Null -6	7, 19,0,9	
	7	8,9,10			
	19				

isYoungWorker

Variable	Partition	Limit	Guess	Final	Explanation

AirlineCustomer.getAge()	-8	-1,0,1	MININT -1 MAXINT +1 8.978 "nine" Null -8	11,21,30,0,18,25	Some values are not included in the final set of values because they are being controlled in the command line interface.
	11	17,18,19			
	21	24,25,26			
	30				
AirlineCustomer.getTravelerType()	STUDENT WORKER		"neither" Null 7	STUDENT WORKER	
AirlineCustomer.getTravelClass()	ECONOMY BUSINESS		"other" 0 null	ECONOMY BUSINESS	
AirlineAirlineCustomer.getFlightsPerYear()	-10	-1,0,1	MININT -1 MAXINT +1 10.86 "eleven" Null -6	2,19, 0,3	
	2	2,3,4			
	19				

isMidIncomeAdult()

Variable	Partition	Limit	Guess	Final	Explanation
AirlineCustomer.getAge()	-8	-1,0,1	MININT -1 MAXINT +1 8.978 "nine" Null -8	21,30, 0, 25	Some values are not included in the final set of values because they are being controlled in the command line interface.
	21	24,25,26			
	30				
AirlineCustomer.getTravelClass()	ECONOMY BUSINESS		"other" 0 null	ECONOMY BUSINESS	
AirlineCustomer.getFlightsPerYear()	-10	-1,0,1	MININT -1 MAXINT +1 10.86 "eleven" Null -6	5, 19 , 0 , 6	
	5	5,6,7			
	19				
AirlineCustomer.getDestination()	EUROPE ASIA AMERICA OTHER		"nowhere" 0 null	EUROPE ASIA AMERICA OTHER	
AirlineCustomer.getIncome()	-200	-1,0,1	MININT -1 MAXINT +1 "million" Null -1	10000, 30000, 1000000, 0, 20000, 35000	

	10000	19999,20000,20001			
	30000	34999,35000,35001			
	1000000				

isHighIncomeAdult()

Variable	Partition	Limit	Guess	Final	Explanation
AirlineCustomer.getAge()	-8	-1,0,1	MININT -1 MAXINT +1 8.978 "nine" Null -8	21, 30 , 0, 25	Some values are not included in the final set of values because they are being controlled in the command line interface.
	21	24,25,26			
	30				
AirlineCustomer.getTravelClass()	ECONOMY BUSINESS		"other" 0 null	ECONOMY BUSINESS	
AirlineCustomer.getFlightsPerYear()	-10	-1,0,1	MININT -1 MAXINT +1 10.86 "eleven" Null -6	5, 19, 0, 6	
	5	5,6,7			
	19				
AirlineCustomer.getDestination()	EUROPE ASIA AMERICA OTHER		"nowhere" 0 null	EUROPE ASIA AMERICA OTHER	
AirlineCustomer.getIncome()	-200	-1,0,1	MININT -1 MAXINT +1 "million" Null -1	30000, 1000000, 0, 35000	
	30000	34999,35000,35001			
	1000000				

getFare()

Variable	Partition	Limit	Guess	Final	Explanation
AirlineCustomer.getAge()	-7	-1,0,1	MININT -1 MAXINT +1 8.978 "nine"	11, 30, 0 , 18	Some values are not included in the final set of values because they are being controlled in
	11	17,18,19			

	30		Null -8		the command line interface.
AirlineCustomer.getFlightsPerYear()	-3	-1,0,1	MININT -1 MAXINT +1 10.86 "eleven" Null -6	4, 10, 0, 6	
	4	5,6,7			
	10				
isStudent()	true false		“1” “yes” null	true false	
isYoungWorker()	true false		“1” “yes” null	true false	
AirlineCustomer.isLivesWithParents()	true false		“1” “yes” null	true false	
isMidIncomeAdult()	true false		“1” “yes” null	true false	
isHighIncomeAdult()	true false		“1” “yes” null	true false	
Customer.isTravelWithChildren()	true false		“1” “yes” null	true false	

5) Calculate the maximum possible number of test cases that could be generated from the test values (combinatorial).

IsStudent(): 6*3*3*4= 216

IsYoungWorker(): 6*3*3*4 = 216

IsMidIncomeAdult(): 4*3*4*4*6=1152

IsHighIncomeAdult(): 4*3*4*4*4=768

GetFare(): 4*4*2*2*2*2*2*2= 1024

6) Define a set of test cases to fulfill each use (each value at least once).

In this section we construct a minimal set of test cases for each method to satisfy the each-use (each-value) criterion. For every test unit, we ensure that every selected test value from Section 4 appears in at least one test case.

IsStudent()

Case	Age	FlightsPerYear	TravelerType	TravelerClass
1	0	0	STUDENT	ECONOMY
2	11	7	WORKER	BUSINESS
3	18	9	STUDENT	ECONOMY
4	21	19	WORKER	BUSINESS
5	25	0	STUDENT	ECONOMY
6	30	7	WORKER	BUSINESS

IsYoungWorker()

Case	Age	FlightsPerYear	TravelerType	TravelerClass
1	0	0	STUDENT	ECONOMY
2	11	2	WORKER	BUSINESS
3	18	3	STUDENT	ECONOMY
4	21	19	WORKER	BUSINESS
5	25	0	STUDENT	ECONOMY
6	30	2	WORKER	BUSINESS

IsMidIncomeAdult()

Case	Age	Income	TravelClass	FlightsPerYear	Destination
1	0	0	STUDENT	0	EUROPE
2	21	10000	WORKER	5	ASIA
3	30	20000	STUDENT	6	AMERICA
4	25	30000	WORKER	19	OTHER
5	30	35000	STUDENT	5	EUROPE
6	21	1000000	WORKER	6	ASIA

IsHighIncomeAdult()

Case	Age	Income	TravelClass	FlightsPerYear	Destination
1	0	0	ECONOMY	0	EUROPE
2	21	30000	BUSINESS	5	ASIA
3	30	35000	ECONOMY	6	AMERICA
4	25	1000000	BUSINESS	19	ASIA
5	30	35000	ECONOMY	5	EUROPE
6	21	1000000	BUSINESS	6	OTHER

GetFare()

For getFare we apply each-use to its direct input fields (age, flightsPerYear, livesWithParents, travelWithChildren). The helper predicates (isStudent, isYoungWorker, etc.) are already tested directly in their own methods.

Case	Age	FlightsPerYear	LivesWithParents	TravelWithChildren
1	0	0	TRUE	TRUE
2	11	4	FALSE	TRUE
3	18	6	TRUE	TRUE
4	30	10	FALSE	FALSE

7) Define test sets to achieve pairwise coverage using the algorithm explained in class. The results can be verified using the PICT program5.

Using the same value domains as in Section 4, we apply a pairwise combination strategy so that every pair of values across any two variables appears in at least one test case.

IsStudent()

Case	Age	FlightsPerYear	TravelerType	TravelClass
1	0	0	STUDENT	ECONOMY
2	0	7	WORKER	BUSINESS
3	11	9	STUDENT	BUSINESS
4	11	19	WORKER	ECONOMY
5	18	0	WORKER	BUSINESS
6	18	7	STUDENT	ECONOMY
7	21	9	WORKER	ECONOMY
8	21	19	STUDENT	BUSINESS
9	25	0	STUDENT	ECONOMY
10	25	7	WORKER	BUSINESS
11	30	0	STUDENT	ECONOMY
12	30	7	WORKER	BUSINESS
13	0	9	STUDENT	ECONOMY
14	0	19	STUDENT	ECONOMY
15	11	0	STUDENT	ECONOMY
16	11	7	STUDENT	ECONOMY
17	18	9	STUDENT	ECONOMY
18	18	19	STUDENT	ECONOMY
19	21	0	STUDENT	ECONOMY
20	21	7	STUDENT	ECONOMY
21	25	9	STUDENT	ECONOMY
22	25	19	STUDENT	ECONOMY
23	30	9	STUDENT	ECONOMY

24	30	19	STUDENT	ECONOMY

IsYoungWorker()

Case	Age	FlightsPerYear	TravelerType	TravelClass
1	0	0	STUDENT	ECONOMY
2	0	2	WORKER	BUSINESS
3	11	3	STUDENT	BUSINESS
4	11	19	WORKER	ECONOMY
5	18	0	WORKER	BUSINESS
6	18	2	STUDENT	ECONOMY
7	21	3	WORKER	ECONOMY
8	21	19	STUDENT	BUSINESS
9	25	0	STUDENT	ECONOMY
10	25	2	WORKER	BUSINESS
11	30	0	STUDENT	ECONOMY
12	30	2	WORKER	BUSINESS
13	0	3	STUDENT	ECONOMY
14	0	19	STUDENT	ECONOMY
15	11	0	STUDENT	ECONOMY
16	11	2	STUDENT	ECONOMY
17	18	3	STUDENT	ECONOMY
18	18	19	STUDENT	ECONOMY
19	21	0	STUDENT	ECONOMY
20	21	2	STUDENT	ECONOMY
21	25	3	STUDENT	ECONOMY
22	25	19	STUDENT	ECONOMY
23	30	3	STUDENT	ECONOMY
24	30	19	STUDENT	ECONOMY

isMidIncomeAdult(AirlineCustomer c)

Case	Age	Income	TravelClass	FlightsPerYear	Destination
1	0	0	ECONOMY	0	EUROPE
2	0	10 000	BUSINESS	5	ASIA
3	21	0	BUSINESS	6	AMERICA
4	21	10 000	ECONOMY	19	OTHER
5	25	20 000	ECONOMY	5	AMERICA
6	25	30 000	BUSINESS	0	OTHER
7	30	20 000	BUSINESS	19	EUROPE
8	30	30 000	ECONOMY	6	ASIA
9	0	35 000	ECONOMY	6	OTHER
10	0	1 000 000	ECONOMY	19	AMERICA
11	21	35 000	BUSINESS	0	ASIA
12	21	1 000 000	BUSINESS	5	EUROPE
13	25	0	ECONOMY	19	ASIA
14	25	10 000	ECONOMY	6	EUROPE

15	30	0	ECONOMY	5	OTHER
16	30	10 000	ECONOMY	0	AMERICA
17	0	20 000	ECONOMY	0	ASIA
18	0	30 000	ECONOMY	5	EUROPE
19	21	20 000	ECONOMY	6	OTHER
20	21	30 000	ECONOMY	19	AMERICA
21	25	35 000	ECONOMY	5	EUROPE
22	25	1 000 000	ECONOMY	0	ASIA
23	30	35 000	ECONOMY	19	AMERICA
24	30	1 000 000	ECONOMY	6	OTHER

IsHighIncomeAdult()

Case	Age	Income	TravelClass	FlightsPerYear	Destination
1	0	0	ECONOMY	0	EUROPE
2	0	30 000	BUSINESS	5	ASIA
3	21	0	BUSINESS	6	AMERICA
4	21	30 000	ECONOMY	19	OTHER
5	25	35 000	ECONOMY	5	AMERICA
6	25	100 0000	BUSINESS	0	OTHER
7	30	35 000	BUSINESS	19	EUROPE
8	30	1 000 000	ECONOMY	6	ASIA
9	0	35 000	ECONOMY	6	OTHER
10	0	1 000 000	ECONOMY	19	AMERICA
11	21	35 000	ECONOMY	0	ASIA
12	21	1 000 000	ECONOMY	5	EUROPE
13	25	0	ECONOMY	19	ASIA
14	25	30 000	ECONOMY	6	EUROPE
15	30	0	ECONOMY	5	OTHER
16	30	30 000	ECONOMY	0	AMERICA

GetFare()

Case	Age	FlightsPerYear	isStudent	isYoungWorker	isMidIncomeAdult	isHighIncomeAdult	livesWithParents	travelWithChildren
1	0	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
2	0	4	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
3	11	6	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE
4	11	10	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE
5	18	0	FALSE	TRUE	TRUE	FALSE	TRUE	TRUE
6	18	4	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE
7	30	6	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE
8	30	10	FALSE	TRUE	FALSE	TRUE	TRUE	TRUE
9	11	0	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE
10	11	4	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
11	0	6	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
12	0	10	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
13	18	6	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

14	18	10	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
15	30	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
16	30	4	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

8) For code sections involving decisions, propose a set of test cases to achieve decision coverage.

Decision coverage requires that every `if` statement in the code evaluates to true at least once and to false at least once. We provide tests where each helper predicate (`isStudent`, `isYoungWorker`, `isMidIncomeAdult`, `isHighIncomeAdult`) is both true and false, and a set of complete `AirlineCustomer` profiles that exercise all branches in `getFare`, including all possible fares and the null case.

Decision coverage for `isStudent`, isYoungWorker, isMidIncomeAdult, isHighIncomeAdult

Test ID	Method	Age	FlightsPerYear	TravelerType	TravelClass	Income	Destination	Expected Result
S1	isStudent	20	9	STUDENT	ECONOMY	0	EUROPE	true
S2	isStudent	20	9	STUDENT	BUSINESS	0	EUROPE	false
Y1	isYoungWorker	23	3	WORKER	ECONOMY	0	EUROPE	true
Y2	isYoungWorker	23	2	WORKER	ECONOMY	0	EUROPE	false
M1	isMidIncomeAdult	30	6	WORKER	ECONOMY	25000	EUROPE	true
M2	isMidIncomeAdult	30	6	WORKER	ECONOMY	15000	EUROPE	false
H1	isHighIncomeAdult	40	6	WORKER	BUSINESS	50000	ASIA	true
H2	isHighIncomeAdult	40	6	WORKER	ECONOMY	50000	ASIA	false

Decision coverage for getFare

Here complete AirlineCustomer profiles that ensure every if in getFare is true at least once and false at least once, and that every fare is produced at least once.

Test ID	Age	FlightsPerYear	isStudent	isYoungWorker	isMidIncomeAdult	isHighIncomeAdult	LivesWithParents	TravelWithChild
G1	15	6	false	false	false	false	false	false
G2	20	9	true	false	false	false	true	false
G3	23	3	false	true	false	false	true	false
G4	23	3	false	true	false	false	false	false
G5	35	6	false	false	true	false	false	true

G6	35	6	false	false	true	false	false	false
G7	40	6	false	false	false	true	false	true
G8	40	6	false	false	false	true	false	false
G9	40	1	false	false	false	true	false	false

9) For code sections involving decisions, propose a set of test cases to achieve MC/DC coverage.

In this section we apply MC/DC to the four helper methods **`isStudent`**, **`isYoungWorker`**, **`isMidIncomeAdult`**, and **`isHighIncomeAdult`**, because these contain the main complex decisions of the system. The `getFare` method is already covered at the decision level in Section 8 and mainly delegates its complex conditions to these helper methods.

IsStudent(c):

Test	Age	TravelerType	TravelClass	Flights	A: age≥18	B: age≤25	C: type=STUDENT	D: class=ECONOMY	E: flights≥9	Result
S1	20	STUDENT	ECONOMY	9	T	T	T	T	T	**true**
S2	17	STUDENT	ECONOMY	9	F	T	T	T	T	false
S3	26	STUDENT	ECONOMY	9	T	F	T	T	T	false
S4	20	WORKER	ECONOMY	9	T	T	F	T	T	false
S5	20	STUDENT	BUSINESS	9	T	T	T	F	T	false
S6	20	STUDENT	ECONOMY	8	T	T	T	T	F	false

MC/DC for isYoungWorker

Test	Age	TravelerType	TravelClass	Flights	F: age≥18	G: age≤25	H: type=WORKER	I: class=ECONOMY	J: flights≥3	Result
Y1	22	WORKER	ECONOMY	3	T	T	T	T	T	**true**
Y2	17	WORKER	ECONOMY	3	F	T	T	T	T	false
Y3	26	WORKER	ECONOMY	3	T	F	T	T	T	false
Y4	22	STUDENT	ECONOMY	3	T	T	F	T	T	false
Y5	22	WORKER	BUSINESS	3	T	T	T	F	T	false
Y6	22	WORKER	ECONOMY	2	T	T	T	T	F	false

IsMidIncome(AirlineCustomer c)

Test	Age	Income	TravelClass	Flights	Destination	K: age>25	L: inc≥20000	M: inc<35000	N: class=ECONOMY	O: flights≥6	P: dest=EUROPE	Result

M1	30	25000	ECONOMY	6	EUROPE	T	T	T	T	T	T	**true**
M2	25	25000	ECONOMY	6	EUROPE	F	T	T	T	T	T	false
M3	30	19000	ECONOMY	6	EUROPE	T	F	T	T	T	T	false
M4	30	36000	ECONOMY	6	EUROPE	T	T	F	T	T	T	false
M5	30	25000	BUSINESS	6	EUROPE	T	T	T	F	T	T	false
M6	30	25000	ECONOMY	5	EUROPE	T	T	T	T	F	T	false
M7	30	25000	ECONOMY	6	ASIA	T	T	T	T	T	F	false

isHighIncomeAdult(AirlineCustomer c)

Test	**Age**	**Income**	**TravelClass**	**Flights**	**Destination**	**A: age>25**	**B: inc≥35000**	**C: class=BUSINESS**	**D: flights≥6**	**E: dest∈{ASIA,AMERICA}**	**Result**
H1	40	50000	BUSINESS	6	ASIA	T	T	T	T	T	**true**
H2	25	50000	BUSINESS	6	ASIA	F	T	T	T	T	false
H3	40	34000	BUSINESS	6	ASIA	T	F	T	T	T	false
H4	40	50000	ECONOMY	6	ASIA	T	T	F	T	T	false
H5	40	50000	BUSINESS	5	ASIA	T	T	T	F	T	false
H6	40	50000	BUSINESS	6	EUROPE	T	T	T	T	F	false

10) Comment on the results regarding the number of test cases obtained in sections 4, 5, and 6: what can be said about the coverage achieved? To what extent does the implementation of the program influence the design and implementation of the test cases??

The maximum number of test cases calculated in section 5 is already significant, although it was calculated using reduced domains from section 4. If we tried to test all combinations, the total number of test cases would clearly be too large to execute and maintain in practice.

In Section 6 we applied the **each-use** (each-value) criterion separately to each method. This gives good input value coverage; all ages, income partitions, classes, destinations and booleans are used at least once, while using only a small number of tests per method.

However, each-use ignores:

- how values combine with each other (interactions), and
- how the code behaves internally (branches / decisions).

So each-use alone is not enough to detect all interaction faults or logical errors, but it is an important first step to ensure no chosen value is forgotten.

In Section 7 we strengthened the input coverage by applying **pairwise testing**. Compared with each-use, the number of tests increases but still remains much lower than full combinatorial coverage. In return, we now cover all **2-way interactions**

The implementation has strong impact on the test design and on the number of test cases.

- The choice of boundary values in the code (age 18 and 25, income 20 000 and 35 000, flights 3, 6 and 9) directly determine the equivalence classes and boundary values we selected in Section 4.
- The seperation of helper methods shaped the structure of sections 6-9
- The use of seperate if statements in the getFare.

The final test cases:

These are the final test cases we implemented, which are both the pairwise and the MC/DC tests.

isStudent(c)

Case	Age	Flights/Year	TravelerType	TravelClass	Condition Result
1	0	0	STUDENT	ECONOMY	false
2	0	7	WORKER	BUSINESS	false
3	11	9	STUDENT	BUSINESS	false
4	11	19	WORKER	ECONOMY	false
5	18	0	WORKER	BUSINESS	false
6	18	7	STUDENT	ECONOMY	false
7	21	9	WORKER	ECONOMY	false
8	21	19	STUDENT	BUSINESS	false
9	25	0	STUDENT	ECONOMY	false

10	25	7	WORKER	BUSINESS	false
11	30	0	STUDENT	ECONOMY	false
12	30	7	WORKER	BUSINESS	false
13	0	9	STUDENT	ECONOMY	false
14	0	19	STUDENT	ECONOMY	false
15	11	0	STUDENT	ECONOMY	false
16	11	7	STUDENT	ECONOMY	false
17	18	9	STUDENT	ECONOMY	true
18	18	19	STUDENT	ECONOMY	true
19	21	0	STUDENT	ECONOMY	false
20	21	7	STUDENT	ECONOMY	false
21	25	9	STUDENT	ECONOMY	true
22	25	19	STUDENT	ECONOMY	true
23	30	9	STUDENT	ECONOMY	false
24	30	19	STUDENT	ECONOMY	false
S1	20	9	STUDENT	ECONOMY	true
S2	17	9	STUDENT	ECONOMY	false
S3	26	9	STUDENT	ECONOMY	false
S4	20	9	WORKER	ECONOMY	false

S5	20	9	STUDENT	BUSINESS	false
S6	20	8	STUDENT	ECONOMY	false

isYoungWorker(c)

Case	Age	Flights/Year	TravelerType	TravelClass	Condition Result
1	0	0	STUDENT	ECONOMY	false
2	0	2	WORKER	BUSINESS	false
3	11	3	STUDENT	BUSINESS	false
4	11	19	WORKER	ECONOMY	false
5	18	0	WORKER	BUSINESS	false
6	18	2	STUDENT	ECONOMY	false
7	21	3	WORKER	ECONOMY	true
8	21	19	STUDENT	BUSINESS	false
9	25	0	STUDENT	ECONOMY	false
10	25	2	WORKER	BUSINESS	false
11	30	0	STUDENT	ECONOMY	false
12	30	2	WORKER	BUSINESS	false
13	0	3	STUDENT	ECONOMY	false
14	0	19	STUDENT	ECONOMY	false

15	11	0	STUDENT	ECONOMY	false
16	11	2	STUDENT	ECONOMY	false
17	18	3	STUDENT	ECONOMY	false
18	18	19	STUDENT	ECONOMY	false
19	21	0	STUDENT	ECONOMY	false
20	21	2	STUDENT	ECONOMY	false
21	25	3	STUDENT	ECONOMY	false
22	25	19	STUDENT	ECONOMY	false
23	30	3	STUDENT	ECONOMY	false
24	30	19	STUDENT	ECONOMY	false
Y1	22	3	WORKER	ECONOMY	true
Y2	17	3	WORKER	ECONOMY	false
Y3	26	3	WORKER	ECONOMY	false
Y4	22	3	STUDENT	ECONOMY	false
Y5	22	3	WORKER	BUSINESS	false
Y6	22	2	WORKER	ECONOMY	false

isMidIncomeAdult(c)

Case	Age	Income	TravelClass	Flights/Year	Destination	Condition Result

1	0	0	ECONOMY	0	EUROPE	false
2	0	10000	BUSINESS	5	ASIA	false
3	21	0	BUSINESS	6	AMERICA	false
4	21	10000	ECONOMY	19	OTHER	false
5	25	20000	ECONOMY	5	AMERICA	false
6	25	30000	BUSINESS	0	OTHER	false
7	30	20000	BUSINESS	19	EUROPE	false
8	30	30000	ECONOMY	6	ASIA	false
9	0	35000	ECONOMY	6	OTHER	false
10	0	1000000	ECONOMY	19	AMERICA	false
11	21	35000	BUSINESS	0	ASIA	false
12	21	1000000	BUSINESS	5	EUROPE	false
13	25	0	ECONOMY	19	ASIA	false
14	25	10000	ECONOMY	6	EUROPE	false
15	30	0	ECONOMY	5	OTHER	false
16	30	10000	ECONOMY	0	AMERICA	false
17	0	20000	ECONOMY	0	ASIA	false
18	0	30000	ECONOMY	5	EUROPE	false
19	21	20000	ECONOMY	6	OTHER	false

20	21	30000	ECONOMY	19	AMERICA	false
21	25	35000	ECONOMY	5	EUROPE	false
22	25	1000000	ECONOMY	0	ASIA	false
23	30	35000	ECONOMY	19	AMERICA	false
24	30	1000000	ECONOMY	6	OTHER	false
M1	30	25000	ECONOMY	6	EUROPE	true
M2	25	25000	ECONOMY	6	EUROPE	false
M3	30	19000	ECONOMY	6	EUROPE	false
M4	30	36000	ECONOMY	6	EUROPE	false
M5	30	25000	BUSINESS	6	EUROPE	false
M6	30	25000	ECONOMY	5	EUROPE	false
M7	30	25000	ECONOMY	6	ASIA	false

isHighIncomeAdult(c)

Case	Age	Income	TravelClass	Flights/Year	Destination	Condition Result
1	0	0	ECONOMY	0	EUROPE	false
2	0	30000	BUSINESS	5	ASIA	false
3	21	0	BUSINESS	6	AMERICA	false
4	21	30000	ECONOMY	19	OTHER	false

5	25	35000	ECONOMY	5	AMERICA	false
6	25	1000000	BUSINESS	0	OTHER	false
7	30	35000	BUSINESS	19	EUROPE	false
8	30	1000000	ECONOMY	6	ASIA	false
9	0	35000	ECONOMY	6	OTHER	false
10	0	1000000	ECONOMY	19	AMERICA	false
11	21	35000	ECONOMY	0	ASIA	false
12	21	1000000	ECONOMY	5	EUROPE	false
13	25	0	ECONOMY	19	ASIA	false
14	25	30000	ECONOMY	6	EUROPE	false
15	30	0	ECONOMY	5	OTHER	false
16	30	30000	ECONOMY	0	AMERICA	false
H1	40	50000	BUSINESS	6	ASIA	true
H2	25	50000	BUSINESS	6	ASIA	false
H3	40	34000	BUSINESS	6	ASIA	false
H4	40	50000	ECONOMY	6	ASIA	false
H5	40	50000	BUSINESS	5	ASIA	false
H6	40	50000	BUSINESS	6	EUROPE	false

getFare(c)

Case	Age	Flights/Year	isStudent	isYoungWorker	isMidIncomeAdult	isHighIncomeAdult	LivesWithParents	TravelWithChildren	Result / Fare
G1	15	6	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	Pajarillo, 10
G2	20	9	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	Gorrión, 15
G3	23	3	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	Travel While You Can, 5
G4	23	3	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	Daring to Leave the Nest, 25
G5	35	6	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	Discover Europe with Your Little Ones, 10
G6	35	6	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	Discover Europe, 15
G7	40	6	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	Discover the World with Your Little Ones, 10
G8	40	6	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	Discover the

									World, 20
G9	40	1	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	null