Part II - Practical Testing – Black Box Testing

Consider the following program specifications. Create the test cases and test data for the Black box test techniques of Equivalence Partitioning and Boundary Value analysis.

railTicket()

A program for rail ticketing returns the level of discount available depending on the age of the traveler that is input: Children under 5 travel free; Children and young adults between the ages of 5 and 16 inclusive receive a 50% discount; those aged 65 and over get a 25% discount. An entry of less than 1 year and greater than 110 years is incorrect and the discount value returned is -1% to reflect the error. In all cases the age must be input as an integer value.

MartialArtsClub()

The specification for a program that computes the appropriate membership fee for a martial arts club is given below. The fee is based on the age of the member only. Membership is not allowed to ages outside the range of 6 to 65.

If the age of the member is from 6 to 18 the Membership fee is €100.
If the age of the member is from 18 to 40 the Membership fee is €200.
If the age of the member is from 40 to 65 the Membership fee is €180.

The program input consists of 1 parameter: int age. The program output is the memberFee value in Euros.

Any invalid input returns a fee of €0. Invalid input is when an age is out of range.

concertTicket()

The specification for a program that computes the ticket cost of a concert at a particular venue is given below. The fee is based on the age of the attendee and their membership status. People outside the age range of 18 to 50 are not allowed to buy a ticket.

If the age of the member is from 18 to 35 the ticket cost is €40 if they are not a member and it is €35 if they are a member
If the age of the member is from 36 to 50 the ticket cost is €65 if they are not a member and it is €55 if they are a member.

The program input consists of 2 parameters: int age and boolean isMember. The program output is the int ticketCost value in Euros. Any invalid input returns a value of -€1.

purchasePrice()

A software program is used by a store to compute the discount it offers on the amount of purchases made by an individual: The purchase price input is an integer.

A purchase amount is in the range of €1 up to €50 there is no discount,
A purchase amount over €50 and up to €200 has a 5% discount,
A purchase amount of €201 and up to €500 has a 10% discount,
A purchase amount of €501 and above has a 15% discount

Additionally, if the customer is a member of the loyalty club they get an added discount of 20% on purchases of value €201 and greater.

If the value entered for the purchase is accidentally entered as a negative number the discount value returned is -1%.

The program input consists of 2 parameters: int purchases and boolean isMember. The program output is the int discount value as a percentage. Any invalid input returns a value of -1%.

StoreDiscounts()

A store in city offers different discounts depending on the purchases made by the individual. In order to test the software program that calculates the discounts, it is possible to identify the "price range" of purchase values that earn the different discounts.

For example,
if a purchase is in the price range of €1 up to and including €50 it has no discounts,
a purchase in the price range of over €50 and up to €200 has a 5% discount, and
purchases of price range €201 and up to €500 have a 10% discount,
and purchases of price range €501 to €2500 have 15% discounts.

No items are sold outside these price ranges and if an incorrect price is entered the software returns a discount value of -1% to indicate that an error case has occurred.

Consider the following program specifications. Create the test cases and test data for the Black box test techniques of Combinational Testing.

Grade()

The program Grade combines an exam and coursework mark into a single grade. The values for exam and coursework are integers. If the exam or coursework mark is less than 50% then the grade returned is a 'Fail'. To pass the course with a 'Pass, C', the student must score between 50% and 60% in the exam, and at least 50% in the coursework. They will pass the course with 'Pass, B', if they score over 60% in the exam and 50% in the coursework. In addition to this, if the average of the exam and the coursework is at least 70%, then they are awarded a 'Pass, A'. Input values that are less than 0 or greater than 100 for either the exam or coursework are invalid and the program will return a message to say 'Marks out of range'.

ComputeInsurance()

A program ComputeInsurance is used by an airline company to automatically assess the level of insurance the customer must pay on their ticket. Each customer can bring one piece of sports equipment and one piece of musical equipment on a flight:
If they bring both sports and music equipment the insurance is €20
If they only bring one piece of equipment only then the insurance is €10
If they bring no equipment then the insurance fee is €5
The program input consists of two boolean variables:
(1) sportsEquipment (2)musicEquipment
The program output is a single variable: insurance

PremiumWithoutMarried()

This program 'CarIns' assesses the cost of a car insurance policy. It takes three inputs of age, gender and marital status. If the age entered is less than 16 or greater than 65 the program returns a premium of zero. The input for gender takes the form of 'M' for male and 'F' for female. If an incorrect value for the gender is entered the program returns a premium of zero. In general a premium is €500. However, if a person is male, under 25 and not married then an extra €1500 is added to the premium. If the person is female and married the premium falls by €200, and if the person is aged between 45 and 65 inclusive the premium falls by €100.

airlineTicket()

An airline offers discounts to passengers depending on their age, the destination of the flight (either domestic or international), the amount of time the flight was paid for in advance, and whether it is scheduled for the busy season or not. It is assumed in the first place that the discount to be offered is 0% and this is adjusted depending on the specific details:

Airline passengers are assigned to three age brackets: Adult which is greater than 16 and up to 100 years, Youth which is classed as being greater than 2 up to 16 years, and infant which is from 0 to 2 years inclusive. The age of a passenger cannot be less than 0 or greater than 100, otherwise a discount value of -1% is output. Discounts are added together to get the total discount value

For international flights, Adult passengers only are offered 15% discount if they travel during outside of the busy season. Any Adult passenger who has an advance reservation before their journey are also offered a discount of 10%.

Youth passengers are given a discount of 10%, for any type of destination. Any Youth passenger who has an advance reservation before their journey are offered a discount of 5%.

Infant passengers (defined as being under two years old) are offered a discount of 80% on domestic flights and are offered a discount of 70% on international flights. They are entitled to get an outside busy season discount of 10% on international flights only. They cannot receive the advanced reservation discount.

checkEquilateral()

A program CheckEquilateral takes three integers as input that represents the lengths of the three sides of a triangle. The program first determines if the three numbers are valid inputs, that is, the length of any sides must not be 0 or a negative number. If not, it will return 'Not a triangle'. Otherwise, it will check whether it is an "Equilateral triangle" (three sides are the same length). Using the Causes and Effects as given below, complete the Truth Table (note: impossible test cases should be eliminated).

Causes	**Effects**
Side1>0	Not a triangle
Side2>0	Equilateral
Side3>0	Not Equilateral
Side1=Side2	
Side2=Side3	
Side1=Side3	

printMessage()

The program "Print message" reads two input characters and, depending on their values, various different actions will occur. Constraints on the two input characters are (1) the first character should be an "A" or a "B", and (2) the second character should be a digit.

If the first character is an "A" or "B" and the second character is a digit, the message “Update file” will appear.

If the first character is incorrect (not an "A" or "B"), the message X must be printed.

If the second character is incorrect (not a digit), the message Y must be printed.

If both the first and the second characters are incorrect, the message XY must be printed.

The Causes and Effects for this program can be written

Causes:
1 - first character is "A"
2 - first character is "B"
3 - second character is a digit

Effects:
4 - message Update file is printed
5 - message X is printed
6 - message Y is printed
7 - message XY is printed

Practical Testing – White Box Testing

Consider the following program code example. Create the test cases and test data for the White box test techniques of (a) Statement Testing, (b) Branch Testing and (c) Path Testing. To make the test cases for each program you must produce a Control Flow Graph (CFG).

taxRelief()

The program renting TaxRelief calculating for the year 2014 of the Revenue Office in Ireland is based on the following specification:

The tax relief is applied to persons aged over 17. If a person is single, aged up to 55, the tax relief is €800, but if he/she is over 55, the tax relief is €1600. If a person is not single (married/widow/with partner, etc.), aged up to 55, the tax relief is €1600, but if he/she is over 55, the tax relief is €3200.

```
1  public int Taxrelief(int age, boolean single)
2  {
3  int rv;
4  if (age<18)
5          rv=0;
6  else if (single)
7                  if (age<=55)
8                          rv=800;
9                  else
10                         rv=1600;
11         else
12                 if (age<=55)
13                         rv=1600;
14                 else
15                         rv=3200;
16 return rv;
17 }
```

memberFee()

The specification of a program to calculate monthly member fee of a Pilates club is as follows: The club is open to persons aged between 16 and 65 only. A Gold member can practice for an unlimited numbers of sessions. A Silver member can take only 2 sessions per week. The basic fee for a Gold member is €70 a month, and €40 for a Silver member. In addition, if the person is a student or aged under 26 or over 60 then they get a discount of €10 off the basic price.

Input parameters:
Age: integer, any input outside of the age ranges are invalid Goldmember: boolean (if they are not a Gold member then they are assumed to be a Silver member by default). Student: boolean.

Output value:
Fee is 0 for any invalid input Fee is 70 if 25<=age<=60 and Gold member and not a student Fee is 40 if 25<=age<=60 and Silver member and not a student

Fee is 60 if Gold member, age<25 or age>60 or student Fee is 30 if Silver member, age<25 or age>60 or student

```
1   public int MemberFee(int age, boolean goldmember, boolean
    student){
2         int fee=0;
3         if ((age<16) || (age>65))
4            System.out.println("This age is outside of the
    membership age range");
5         else
6               if (goldmember)
7                     if ((age<25) || (age>60) || (student))
8                           fee=60;
9                     else
10                          fee=70;
11              else
12                    if ((age<25) || (age>60) || (student))
13                          fee=30;
14                    else
15                          fee=40;
16              return fee;
17        }
```

phoneIns()

The program 'PhoneIns' computes the cost of a smartphone insurance policy and outputs a value for the premium as denoted by *p*. It takes two inputs of integer *age* and Char *OS* (Operating System) type.

If the age entered is less than 16 or greater than 99 the program returns a premium of zero, *p*=0.
The input for OS takes the form of 'I' for IoS, 'A' for Android, and 'W' for Windows. If an incorrect value for the OS is entered, the program returns *p*=0.

In general the insurance premium is €50, *p*=50.

However, if a person has an iPhone and is under 25 then an extra €25 is added to the premium, *p*=75.

If the person is aged between 40 and 60 (inclusive) and they have an Android phone the premium falls by €10, *p*=40.

If the person is aged between 61 and 65 inclusive the premium falls by €5, *p*=45.

Line No.	Code
1	`public int phoneIns (int age, char OS) {`
2	`int p;`
3	`if ((age<16) || (age>99) || (OS!='I' &&OS!='A' &&  OS!='W'))`
4	`p=0;`
5	`else {`
6	`p=50;`
7	`if ((age<25) && (OS=='I'))`
8	`p += 25;`
9	`else {`
10	`if ((age>=40) && (age<=60) && OS=='A')`
11	`p -= 10;`
12	`else if ((age>=61) && (age<=65))`
13	`p -= 5;`
14	`}`
15	`}`
16	`return p;`
17	`}`

CA()

The program CA combines the marks for two assessments ‘assign1’ and ‘assign2’ mark into a single grade. The marks value for both assignments is an integer. If the marks for either assignment is less than 50% then the grade returned is a ‘Fail’. To pass the CA with a ‘Pass, C’, the student must score between 50% and 60% (inclusive) in ‘assign1’ or ‘assign2’. They will pass the course with at least a ‘Pass, B’, if they score greater than 60% in both ‘assign1’ and ‘assign2’. In addition to this, if the average of ‘assign1’ and ‘assign2’ is 80% or more, then they are awarded a ‘Pass, A’. An input values that is less than 0 or greater than 100 for either assignment is invalid, and the program will return a message: ‘Marks out of range’.

Line No.	Code
1	`public static String CA (int assign1, int assign2) {`
2	`String result="null";`
3	`long average = Math.round((assign1+assign2)/2);`
4	`if ((assign1<0) || (assign1>100) || (assign2<0) || (assign2>100))`
5	`result="Marks out of range";`
6	`else {`
7	`if ((assign1<50) || (assign2<50)) {`
8	`result="Fail";`
9	`}`
10	`else if ((assign1>=50 && assign1<=60) || (assign2>=50 && (assign2<=60) {`
11	`result="Pass,C";`
12	`}`
13	`else if (average >= 80) {`
14	`result="Pass,A";`
15	`}`
16	`else {`
17	`result="Pass,B";`
18	`}`
19	`}`
20	`return result;`
21	`}`

memberFee()

A sport centre in a university offers a range of sport activities to its [10 marks] staffs and students. A program calculates the annual fee as follows:
If a staff registers as a single member then the price is €450 a year,
as a couple then the price is €850 a year,
as a family then the price is €1000 a year.
The annual fee for student is €300 and is just for only one person.

Specification
Input: type (char): 'A' (staff) or 'S' (student)
scheme (integer): 1(single), 2 (couple) , 4 (family) Output fee as the following

- Staff single: €450
- Staff couple: €850
- Staff family: €1000
- Student: €300

Any invalid input value will return 0.

```
1  public static int memberFee(char type, int scheme)
2  {
3    int fee;
4    if (((type !='A') && (type !='S')) ||
   ((scheme != 1) && (scheme !=2) && (scheme != 4)))
5          fee=0;
6   else
7        if (type=='A')
8              if (scheme==1)
9                    fee=450;
10             else if (scheme==2)
11                   fee=850;
12                 else
13                   fee=1000;
14       else
15                   if (scheme==1)
16                         fee=300;
17                   else
18                         fee=0;
19             return fee;
20       }
```

ComputeInsurance()

A program ComputeInsurance is used by an airline company to automatically assess the level of insurance the customer must pay on their ticket. Each customer can bring one piece of sports equipment and one piece of musical equipment on a flight:

If they bring both sports and music equipment the insurance is €20

If they only bring one piece of equipment only then the insurance is €10

If they bring no equipment then the insurance fee is €5

The program input consists of two boolean variables:

(1) sportsEquipment

(1) musicEquipment

The program output is a single variable: insurance

Line No.	Code
1)	**public int** ComputeInsurance(**boolean** sportsEquipment, **boolean** musicEquipment)
2)	{
3)	**int** insurance;
4)	**If** (sportsEquipment == **true** && musicEquipment == **true**)
5)	insurance = 20;
6)	**else if** ((sportsEquipment == **true** && musicEquipment == **false**) || (sportsEquipment == **false** && musicEquipment == **true**))
7)	insurance = 10;
8)	**Else**
9)	insurance = 5;
10)	**return** insurance;
11)	}

orderScreening()

OrderScreening() is a program for screening orders received from customers based on three values: quantity of the order, credit-worthy status of customer, and the inventory quantity. The program output is a string. It depends on the values of the three parameters, the output will be: "Accept order", "Reject order", or "Defer order".

The calculation is as follows:

- If the ordered quantity is smaller than or equal to 1000 items (the maximum limit), and the customer is credit-worthy, and the inventory is larger than or equal to the ordered quantity, then the order will be accepted.
- If the ordered quantity is smaller than or equal to 1000 items (the maximum limit), and the customer is credit-worthy, but the inventory is less than the ordered quantity, then the order will be deferred.
- All the other cases, the orders will be rejected.

```
1   public static String OrderScreening(int quantity,
    boolean credit_worthy, int inventory)
2   {
3           String output;
4           if ((quantity<=1000) && (credit_worthy))
5                   if (quantity<=inventory)
6                           output="Accept order";
7                   else
8                           output="Defer order";
9           else
10                  output="Reject order";
11          return output;
12  }
```

computeDiscount()

A program ComputeDiscount is used by an Adventure Trip Association to automatically calculate the discount for participants for each trip. The trips are opened for every one (i.e. members and non-members of the association) from the age 18 to 65. Persons are out of range of those ages, are not allowed to take trips.
There is no discount (i.e. discount is 0) for non-member participants of any ages in that range.
For member participants aged between 18 and 25 or between 56 and 65 and if he/she is a member of the association for at least full 3 years, the discount is 25%, and if he/she is a member less than 3 years the discount is 15%.
For member participants not in those ages and if he/she is a member of the association for at least full 3 years, the discount is 20%, and the discount is 10% for members less than 3 years.

The program input consists of 3 parameters:
int age, boolean member, int memberyear

The program output is 1 return value: double discount
- discount is 0 for any invalid input
- discount is 0 for non member
- discount is 0.25 for members aged between 18 and 25, or between 56 and 65 and at least 3 years member
- discount is 0.15 for members aged between 18 and 25 or between 56 and 65 and less than 3 years member
- discount is 0.2 for member aged between 26 and 55 and at least 3 years member
- discount is 0.1 for member aged between 26 and 55 and at less than 3 years member.

```
1   public double ComputeDiscount(int age, boolean
    member, int memberyear)
2   {
3   double discount=0;
4   if ((age<18) || (age>65) || (memberyear< 0))
5   {
6           System.out.println("Not valid input");
7   } else
8           if (!member)
9                   discount=0;
10          else
11                  if  ((age<=25) || (age>=56))
12                          if (memberyear>=3)
13                                  discount=0.25;
14                          else
15                                  discount=0.15;
16                  else
17                          if (memberyear>=3)
18                                  discount=0.2;
19                          else
20                                  discount=0.1;
21  return discount;
22  }
```

Grade()

The program Grade combines an exam and coursework mark into a single grade. The values for exam and coursework are integers. If the exam or coursework mark is less than 50% then the grade returned is a 'Fail'. To pass the course with a 'Pass, C', the student must score between 50% and 60% in the exam, and at least 50% in the coursework. They will pass the course with 'Pass, B', if they score over 60% in the exam and 50% in the coursework. In addition to this, if the average of the exam and the coursework is at least 70%, then they are awarded a 'Pass, A'. Input values that are less than 0 or greater than 100 for either the exam or coursework are invalid and the program will return a message to say 'Marks out of range'.

Program Grade

Line No.	Code
1)	`public static String Grade (int exam, int course) {`
2)	`String result="null";`
3)	`long average;`
4)	`average = Math.round((exam+course)/2);`
5)	`if ( (exam<0) || (exam>100) || (course<0) || (course>100) )`
6)	`result="Marks out of range";`
7)	`else {`
8)	`if ( (exam<50) || (course<50)) {`
9)	`result="Fail";`
10)	`}`
11)	`else if (exam < 60) {`
12)	`result="Pass,C";`
13)	`}`
14)	`else if ( average >= 70) {`
15)	`result="Pass,A";`
16)	`}`
17)	`else {`
18)	`result="Pass,B";`
19)	`}`
20)	`}`
21)	`return result;`
22)	`}`

checkContainer()

A software program checkContainer() determines if a user-specified container size can carry a load depending on the weight or volume of the input load. A small ($1m^3$ volume) container can hold a load with a maximum weight of 1000kg, a medium ($10m^3$) container can hold a maximum of 5000kg, and a large ($20m^3$) container can hold a maximum of 10000kg.

Specification:

Input: size: character 'S', 'M', 'L' (container size)

weight: integer number (input weight is in kg indicates the weight of the load, weight must be a positive number)

volume: integer number (the volume of the load to be carried in cubic metres, volume must be a positive number)

Output:

Return false if invalid inputs or the container cannot hold the load
Return true if the container can hold the load

The source code for checkContainer() is given below.

```
1  public boolean checkContainer(char size, int weight, int volume)
2  {
3  boolean ok = true;
4  if ((weight<=0)||(volume<=0))
5  ok=false;
6  else if ((size=='S') && ((weight>1000)||(volume>1)))
7  ok=false;
8  else if ((size=='M') && ((weight>5000)||(volume>10)))
9  ok=false;
10 else if ((size=='L') && ((weight>10000) || (volume>20)))
11 ok=false;
12 return ok;
13 }
```

checkTheDate()

The CheckDate program can check any date from the 1/1/1800 until the 31/12/2399 to ensure it is valid. A valid number between 1 and 12 must be entered for the month. The day in the month is also checked. In particular, it will check that if the day is at the end of the month it is correct: if the month has 31, 30 days, 28 days or 29 days if it is a leap year. The program returns true is the date is correct and false otherwise.

Line No.	Code
1)	`public boolean CheckTheDate(int day,int month, int year){`
2)	`boolean date=true;`
3)	`if ((year>2400) || (year<1800) || (month<1) || (month>12) || (day<1) || (day>31))`
4)	`date=false;`
5)	`else if (((month==4) || (month==6) || (month==9) || (month==11)) && (day>30))`
6)	`date=false;`
7)	`else if ((month==2) && (day>29))`
8)	`date=false;`
9)	`else if ((month==2) && (day==29)) {`
10)	`if ((year%100==0 && (year%4!=0 || year%400!=0)) || (year%4!=0))`
11)	`date=false;`
12)	`}`
13)	`return date;`
14)	`}`

airlineSeatReservation()

Specification
A program for airline seat reservation takes two inputs: (1) the number of seats that are free and (2) the number of seats required.

If the number of seats required is less than or equal to the number of free seats then the program should return true. If this is not the case the program should return false. Additionally, if either of the inputs is invalid the program should return false. The smallest valid number for the number of free seats is 0 and the smallest valid number for the number of seats required is 1. The maximum valid number for both of these parameters is the total number of seats, which can be assumed to be 50.

```
(1) public static boolean seatsAvailable(int freeSeats, int seatsRequired) {
(2)    boolean rv=false;
(3)  if ( (freeSets>=0) && (seatsRequired>=1) &&  (seatsRequired<=freeSeats))
(4)       rv=true
(5)    return rv;
(6) }
```

engineService()

The program engineService determines when a service is needed for a car. It has three integer inputs: miles; months; and runhours. The output ServiceNeeded is either true or false. Engine maintenance is required if it has been running more than 20,000 miles, or at least 12 months, since the last service. However, a service is also required if the engine has run for more than 1,000 hours and it is more than 15,000 miles since the last service. Checking is carried out to prevent the entry of zero or negative numbers for any of the three input parameters.

(1)	`public static boolean engineService(int miles,int months,int run_hours) {`
(2)	`boolean ServiceNeeded=false;`
(3)	`if (miles<=0 || months<=0 || run_hours<=0)`
(4)	`ServiceNeeded=false;`
(5)	`else {`
(6)	`if (miles>20000)`
(7)	`ServiceNeeded=true;`
(8)	`else if (months>=12)`
(9)	`ServiceNeeded=true;`
(10)	`else if(run_hours>1000 && miles>15000)`
(11)	`ServiceNeeded=true;`
(12)	}
(13)	return ServiceNeeded;
(14)	}