ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ:
ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ημερομηνία Παράδοσης: 27 Φεβρουάριος 2022

ΠΟΙΟΤΗΤΑ ΚΑΙ ΑΞΙΟΠΙΣΤΙΑ ΛΟΓΙΣΜΙΚΟΥ 2021-22 ΕΡΓΑΣΙΑ ΕΡΓΑΣΤΗΡΙΟΥ

ΑΝΑΠΤΥΞΗ ΒΙΒΛΙΟΘΗΚΗΣ ΣΥΝΑΡΤΗΣΕΩΝ (DLL) ΓΙΑ ΤΗΝ ΔΙΑΧΕΙΡΙΣΗ ΜΙΣΘΟΔΟΣΙΑΣ

Υπεύθυνος Εργαστηρίου: Ακριβή Κρούσκα

Βελάσκο Πάολα	**ΑΜ:161020**	**Εξάμηνο 11ο**	**(ΠΑΔΑ)**
Γιαλαμάς Αθανάσιος	**ΑΜ:141160**	**Εξάμηνο 15ο**	**(ΠΑΔΑ)**
Χριστουλάκης Γεράσιμος	**ΑΜ:151140**	**Εξάμηνο 13ο**	**(ΤΕΙ)**

ΠΕΡΙΕΧΟΜΕΝΑ

ΣΚΟΠΟΣ ΕΡΓΑΣΙΑΣ

Το έγγραφο αυτό έχει ως σκοπό την παρουσίαση της τελικής εργασίας – εξέτασης που ανατέθηκε στο ακαδημαϊκό έτος 2021-22 για το εργαστηριακό μέρος του μαθήματος «Ποιότητα και Αξιοπιστία Λογισμικού». Υπεύθυνη εργασίας: κα Κρούσκα Ακριβή και μέλη της ομάδας: Βελάσκο Πάολα, Γιαλαμάς Αθανάσιος και Χριστουλάκης Γεράσιμος.

Στη συγκεκριμένη εργασία, καλούμαστε να αναπτύξουμε και να υλοποιήσουμε μια βιβλιοθήκη συναρτήσεων για τη διαχείριση μισθοδοσίας – DLL με όνομα SalaryLib.

Ως ομάδα επιλέξαμε να χρησιμοποιήσουμε τα εργαλεία

- της Microsoft Visual Studio 2022 και
- τη γλώσσα προγραμματισμού C#.

Ειδικότερα, μας έχει ανατεθεί να υλοποιήσουμε διάφορες συναρτήσεις και δομές δεδομένων που είναι απαραίτητες στο σύστημα για τη διαχείριση μισθοδοσίας. Για κάθε συνάρτηση είμαστε υποχρεωμένοι εκτός από τον κώδικά τους, να δημιουργήσουμε και τους αντίστοιχους πίνακες ελέγχου και να αναπτύξουμε μονάδες ελέγχου, γνωστά και ως unit tests. Τέλος, πρέπει να επισημάνουμε στην εργασία διάφορες αναφορές ελέγχου για κάθε περίπτωση και σφάλματα μαζί με ενδεικτικά αποτελέσματα μέσω των screenshot, αφού τρέξουμε το πρόγραμμα και τα tests.

ΕΚΠΟΝΗΣΗ ΕΡΓΑΣΙΑΣ

ΜΕΘΟΔΟΥΣ/ΣΥΝΑΡΤΗΣΕΙΣ

1. *bool validEMAIL(string email)*

Η συνάρτηση αυτή δέχεται ως παράμετρο ένα κείμενο και επιστρέφει τιμή true ή false αν η παράμετρος αντιστοιχεί σε email ή όχι, αντίστοιχα.

Α. Κώδικας

```
1.  public bool validEMAIL(string email) {
2.      // Έλεγχος αν είναι άδειο το string
3.      if (string.IsNullOrWhiteSpace(email)) {
4.          return false;
5.      }
6.
7.      var trimmedEmail = email.Trim();
8.
9.      if (trimmedEmail.EndsWith(".")) {
10.         return false;
11.     }
12.     try {
13.         var addr = new System.Net.Mail.MailAddress(email);
14.         return addr.Address == trimmedEmail;
15.     } catch {
16.         return false;
17.     }
18. }
```

Β. Πίνακας – περιπτώσεις ελέγχου

Περ.Ελ.	**Δοκιμαστικά Δεδ. Εισόδου**	**Αναμ. Αποτ.**
#id	**email**	**flag**
1	cs161020@uniwa.gr	true
2	paolavlsc170698@gmail.com	true
3	xristoulakis_02makis@yahoo.org	true
4	onle!234@meil.me	true
5	.@iana.org	false
6	Ima Fool@iana.org	false
7	2343245#@ok.gr	true
8	email@example.com	true
9	email@123.123.123.123	true
10	_____@example.com	true
11	email@example.co.jp	true
12	.first.last@iana.org	false

Γ. Unit Tests

```
1.  [TestMethod]
2.          public void TestValidEmail()
3.          {
4.              //Δημιουργία ενός αντικειμένου της κλάσης του dll που θέλουμε τα τεστάρουμε
5.              SalaryLib.SalaryLib salaryLib = new SalaryLib.SalaryLib();
6.
7.              //Δημιουργία Περιπτώσεων Ελέγχου
8.              object[,] testcases =
9.              {
10.                 {1, true, "cs161020@uniwa.gr"},
11.                 {2, true, "paolavlsc170698@gmail.com"},
12.                 {3, true, "xristoulakis_02makis@yahoo.org"},
13.                 {4, false,"onle!234@meil.me"},
14.                 {5, true, ".@iana.org"},
15.                 {6, false, "Ima Fool@iana.org"},
16.                 {7, true, "2343245#@ok.gr"},
17.                 {8, true, "email@example.com"},
18.                 {9, false, "email@123.123.123.123"},
19.                 {10, true, "_____________@example.com"},
20.                 {11, false, "email@example.co.jp"},
21.                 {12, false, ".first.last@iana.org"}
22.             };
23.
24.             //Αρχικοποίηση δείκτη περιπτώσεων ελέγχου
25.             int i = 0;
26.             bool failed = false;
27.
28.             //Προσπέλαση και εκτέλεση περιπτώσεων ελέγχου
29.             for (i = 0; i < testcases.GetLength(0); i++)
30.             //Για κάθε περίπτωση ελέγχου, δηλαδή για κάθε γραμμή i του πίνακα testcases
31.             {
32.                 try
33.                 {
34.                     //Καλούμε την Assert.AreEqual δίνοντας ως παραμέτρους τα στοιχεία της
    περίπτωσης ελέγχου,
35.                     //δηλαδή τα αντίστοιχα στοιχεία της γραμμής i του πίνακα testcases
36.                     Assert.AreEqual((bool)testcases[i, 1],
    salaryLib.validEMAIL((string)testcases[i, 2]));
37.                 }
38.                 catch (Exception e)
39.                 {
40.                     //Απέτυχε η περίπτωση ελέγχου
41.                     failed = true;
42.                     //Καταγράφουμε την περίπτωση ελέγχου που απέτυχε
43.                     Console.WriteLine("Failed Test Case: {0}\n \t Reason: {1} ",
    (int)testcases[i, 0], e.Message);
44.                 };
45.             };
46.
47.             //Στην περίπτωση που κάποια περίπτωση ελέγχου απέτυχε, πέταξε εξαίρεση.
48.             if (failed) Assert.Fail();
49.         }
```

Δ. Αναφορές Ελέγχου

Αναφορά σφάλματος #1

```
Failed Test Case: 2
      Reason: Assert.AreEqual failed. Expected:<False>. Actual:<True>.
```

- **Αναγνωριστικό Σφάλματος (αποτελούμενο από ένα λεκτικό αναγνωριστικό του σφάλματος και έναν αύξοντα αριθμό)**

```
Failed Test Case: 2
```

- **Όνομα μεθόδους ελέγχου (test method) που απέτυχε**

```
TestValidEmail
```

- **Περιγραφή ελέγχου που πραγματοποιήθηκε (δηλαδή της test method)**

```
Συγκρίνει αν αυτό που επιστρέφει η συνάρτηση ValidEMAIL, είναι ίδια με
αυτή που αναμένουμε.
```

- **Αναφορά σφάλματος που παρουσιάστηκε**

```
Reason: Assert.AreEqual failed. Expected:<False>. Actual:<True>.
```

Αναφορά σφάλματος #2

```
Failed Test Case: 5
      Reason: Assert.AreEqual failed. Expected:<True>. Actual:<False>.
```

- **Αναγνωριστικό Σφάλματος**

```
Failed Test Case: 5
```

- Όνομα μεθόδους ελέγχου που απέτυχε

```
TestValidEmail
```

- **Περιγραφή ελέγχου που πραγματοποιήθηκε (δηλαδή της test method)**

```
Αναμενόμενη τιμή από το χρήστη: False
Τιμή που έβγαλε το πρόγραμμα: True
```

- **Αναφορά σφάλματος που παρουσιάστηκε**

```
Reason: Assert.AreEqual failed. Expected:<False>. Actual:<True>.
```

Αναφορά σφάλματος #3

```
Failed Test Case: 8
      Reason: Assert.AreEqual failed. Expected:<True>. Actual:<False>.
```

- **Αναγνωριστικό Σφάλματος**

```
Failed Test Case: 8
```

- **Όνομα μεθόδους ελέγχου που απέτυχε**

```
TestValidEmail
```

- **Περιγραφή ελέγχου που πραγματοποιήθηκε (δηλαδή της test method)**

```
Αναμενόμενη τιμή από το χρήστη: True
Τιμή που έβγαλε το πρόγραμμα: False
```

- **Αναφορά σφάλματος που παρουσιάστηκε**

```
Reason: Assert.AreEqual failed. Expected:<True>. Actual:<False>.
```

Αναφορά σφάλματος #4

```
Failed Test Case: 11
      Reason: Assert.AreEqual failed. Expected:<False>. Actual:<True>.
```

- **Αναγνωριστικό Σφάλματος**

```
Failed Test Case: 11
```

- **Όνομα μεθόδους ελέγχου που απέτυχε**

```
TestValidEmail
```

- **Περιγραφή ελέγχου που πραγματοποιήθηκε (δηλαδή της test method)**

```
Αναμενόμενη τιμή από το χρήστη: False
Τιμή που έβγαλε το πρόγραμμα: True
```

- **Αναφορά σφάλματος που παρουσιάστηκε**

```
Reason: Assert.AreEqual failed. Expected:<False>. Actual:<True>.
```

Αναφορά σφάλματος #5

```
Failed Test Case: 12
      Reason: Assert.AreEqual failed. Expected:<True>. Actual:<False>.
```

- **Αναγνωριστικό Σφάλματος**

```
Failed Test Case: 12
```

- **Όνομα μεθόδους ελέγχου που απέτυχε**

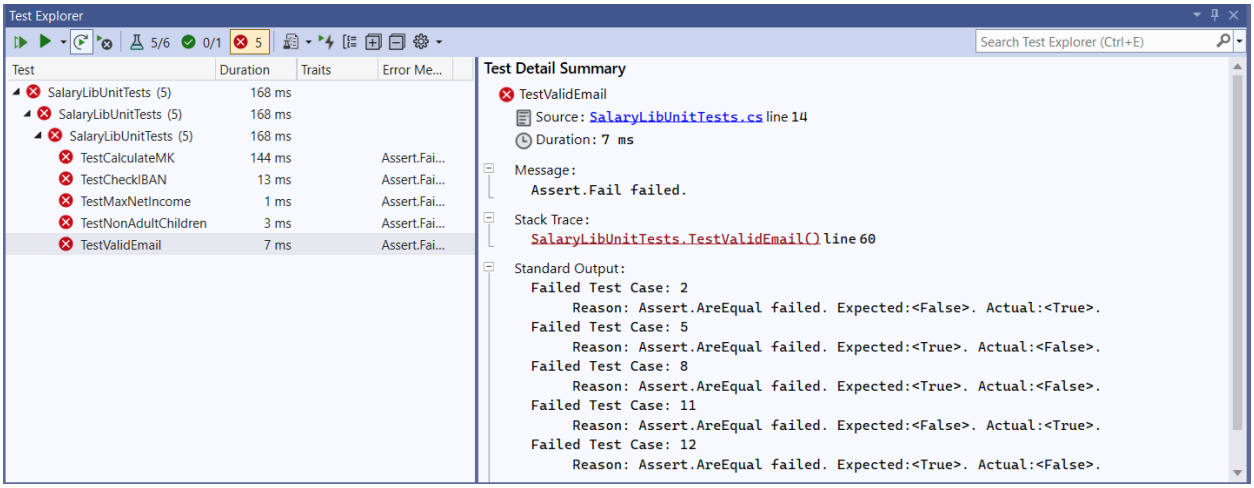
```
TestValidEmail
```

- **Περιγραφή ελέγχου που πραγματοποιήθηκε (δηλαδή της test method)**

```
Αναμενόμενη τιμή από το χρήστη: True
Τιμή που έβγαλε το πρόγραμμα: False
```

- **Αναφορά σφάλματος που παρουσιάστηκε**

```
Reason: Assert.AreEqual failed. Expected:<True>. Actual:<False>.
```



2. *checkIBAN(string IBAN, ref bool validIBAN, ref string bank)*

Η μέθοδος αυτή δέχεται ως κείμενο τον αριθμό IBAN και επιστρέφει αν είναι έγκυρος και την τράπεζα που ανήκει ο λογαριασμός. Τα αποτελέσματα αυτά επιστρέφονται μέσω των αντίστοιχων by ref παραμέτρων που δέχεται. Η υπηρεσία δέχεται μόνο IBAN ελληνικών τραπεζών (επιλέξτε 3 τράπεζες που θα υποστηρίζει το σύστημά σας).

Α. Κώδικας

```
1.  public void CheckIBAN(string IBAN, ref bool validIBAN, ref string bank) {
2.      string iban = IBAN;
3.      string ibanCountryCode = null;
4.      string ibanCodeCheck = null;
5.      string bankCode = null;
6.      string ibanProcessed = null;
7.      BigInteger ibanNumber = 0;
8.      BigInteger result = 0;
9.
10.
11.     // Έλεγχος αν είναι άδειο το string
12.     if (string.IsNullOrWhiteSpace(iban)) {
13.         validIBAN = false;
14.     } else {
15.         // Έλεγχος αν string έχει μήκος 27
16.         if (iban.Length == 27) {
17.             // Έλεγχος αν ανήκει σε ελληνική τράπεζα
18.             ibanCountryCode = iban.Substring(0, 2);; // Λήψη χαρακτήρων των GR
19.
20.             // έλεγχος αν είναι GR
21.             if (!ibanCountryCode.Equals("GR")) {
22.                 validIBAN = false;
23.             } else {
24.                 // Μετατροπή το GR σε αριθμούς 1627
25.                 ibanCountryCode = "1627";
26.
27.                 ibanCodeCheck = iban.Substring(2, 2); // Λήψη χαρακτήρων για ψηφία ελέγχου
28.
```

```
29.                 // επεξεργασία IBAN για έλεγχο
30.                 ibanProcessed = iban.Substring(4) + ibanCountryCode + ibanCodeCheck;
31.
32.                 // Μετατροπή string σε int
33.                 try {
34.                     ibanNumber = BigInteger.Parse(ibanProcessed);
35.                 } catch (FormatException) {
36.                     Console.WriteLine("Unable to convert the string '{0}' to a BigInteger
    value.", ibanProcessed);
37.                 }
38.
39.                 //  Διαίρεση με το 97 και το υπόλοιπο πρέπει να είναι 1
40.                 result = ibanNumber % 97;
41.
42.                 //  Διαίρεση με το 97 και το υπόλοιπο πρέπει να είναι 1
43.                 result = ibanNumber % 97;
44.                 if (result != 1) {
45.                     validIBAN = false;
46.                 } else {
47.                     validIBAN = true;
48.
49.                     // Εύρεση σε ποια τράπεζα ανήκει.
50.                     bankCode = iban.Substring(4, 3); // Λήψη χαρακτήρων για κωδικό τράπεζας
51.
52.                     // alpha bank: 014
53.                     if (bankCode.Equals("014")) {
54.                         bank = "Alpha Bank";
55.                     } // national bank of Greece: 011
56.                     else if (bankCode.Equals("011")) {
57.                         bank = "National Bank of Greece";
58.                     } // eurobank
59.                     else if (bankCode.Equals("026")) {
60.                         bank = "Eurobank";
61.                     } else {
62.                         bank = "N/A";
63.                     }
64.                 }
65.             }
66.         } else {
67.             validIBAN = false;
68.         }
69.     }
70. }
71.
```

Β. Πίνακας – περιπτώσεις ελέγχου

Περ.Ελ.	**Δοκιμαστικά Δεδ. Εισόδου**	**Αναμ. Αποτ.**	
#id	**IBAN**	**validIBAN**	**bank**
1	GR5002602070000610201620365	true	Eurobank
2	GR5002635235454367544364536	false	""
3	GR1601101250000000012300695	true	National Bank of Greece
4	GR5601453552841349465656353	true	Alpha Bank
5	GR5301146556946221684885386	true	National Bank of Greece
6	GR2001165367252351725889763	false	National Bank of Greece
7	GR2601466292141176874543142	true	Alpha Bank
8	EL4525342592358235	true	Eurobank
9	GR2901443615399651293811729	true	Alpha Bank

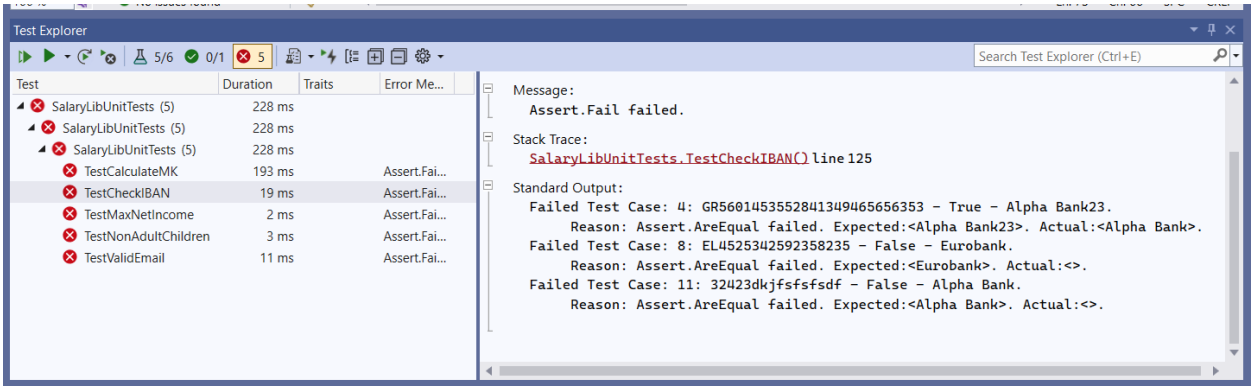
10	GR2101739757567455957413767	false	""
11	32423dkjfsfsfsdf	false	Alpha Bank

Γ. Unit Tests

```
1.  [TestMethod]
2.          public void TestCheckIBAN()
3.          {
4.              // string IBAN, ref bool validIBAN, ref string bank
5.
6.              //Δημιουργία ενός αντικειμένου της κλάσης του dll που θέλουμε τα τεστάρουμε
7.              SalaryLib.SalaryLib salaryLib = new SalaryLib.SalaryLib();
8.
9.              //Δημιουργία Περιπτώσεων Ελέγχου
10.             object[,] testcases =
11.             {
12.                 {1, "GR5002602070000610201620365",true, "Alpha Bank"},
13.                 {2, "GR5002602070000610201620365", false, ""},
14.                 {3,"GR5002602070000610201620365", false, ""},
15.                 {4,"GR5002635235454367544364536",false, ""},
16.                 {5,"GR1601101250000000012300695", true, "National Bank of Greece"},
17.                 {6,"GR5601453552841349465656353", true, "Eurobank" },
18.                 {7,"GR5301146556946221684885386", true, "Alpha Bank" },
19.                 {8,"GR2001165367252351725889763",false, "National Bank of Greece" },
20.                 {9,"GR2601466292141176874543142",true, "Alpha Bank" },
21.                 {10,"EL4525342592358235",true,"Eurobank" },
22.                 {11,"GR2901443615399651293811729", true, "Alpha Bank"},
23.                 {12,"GR2101739757567455957413767", false, "Eurobank" },
24.                 {13,"32423dkjfsfsfsdf",true,"alpha bank" }
25.             };
26.
27.
28.             //Αρχικοποίηση δείκτη περιπτώσεων ελέγχου
29.             int i = 0;
30.             bool failed = false;
31.
32.             //Προσπέλαση και εκτέλεση περιπτώσεων ελέγχου
33.             for (i = 0; i < testcases.GetLength(0); i++)
34.             //Για κάθε περίπτωση ελέγχου, δηλαδή για κάθε γραμμή i του πίνακα testcases
35.             {
36.                 try
37.                 {
38.                     //Δήλωση ref μεταβλητών μεθόδου,
39.                     //προκειμένου να αποθηκευτούν οι τιμές που θα επιστρέψει η μέθοδος
40.                     bool ValidIBAN = false;
41.                     string bankName = "";
42.
43.                     //Καλούμε την μέθοδο που θέλουμε να εξετάσουμε δίνοντας ως παραμέτρους
    τα
44.                     //στοιχεία της περίπτωσης ελέγχου,
45.                     //δηλαδή τα αντίστοιχα στοιχεία της γραμμής i του πίνακα testcases,
46.                     //και ως ref τις μεταβλητές που δηλώσαμε
47.                     salaryLib.CheckIBAN((string)testcases[i, 1], ref ValidIBAN, ref
    bankName);
48.
49.                     //Καλούμε την Assert.AreEqual δίνοντας ως παραμέτρους τις τιμές
50.                     //που θέλουμε να συγκρίνουμε
51.                     Assert.AreEqual((bool)testcases[i, 2], ValidIBAN);
52.                     Assert.AreEqual((string)testcases[i, 3], bankName);
53.                 }
54.                 catch (Exception e)
```

```
55.                 {
56.                     //Απέτυχε η περίπτωση ελέγχου
57.                     failed = true;
58.                     //Καταγράφουμε την περίπτωση ελέγχου που απέτυχε
59.                     Console.WriteLine("Failed Test Case: {0}: {1} - {2} - {3}. \n \t Reason:
    {4} ", (int)testcases[i, 0], (string)testcases[i, 1], (bool)testcases[i, 2],
    (string)testcases[i, 3], e.Message);
60.                 };
61.             };
62.
63.             //Στην περίπτωση που κάποια περίπτωση ελέγχου απέτυχε, πέταξε εξαίρεση.
64.             if (failed) Assert.Fail();
65.         }
```

Δ. Αναφορές Ελέγχου



3. calculateSalary(Employee emplX, ref double grossSalary, ref double netIncome)

Η μέθοδος αυτή δέχεται ως παραμέτρους τα στοιχεία ενός υπαλλήλου, και επιστρέφει τις μικτές και καθαρές αποδοχές του. Τα αποτελέσματα αυτά επιστρέφονται μέσω των αντίστοιχων by ref παραμέτρων που δέχεται. Ο υπολογισμός της μισθοδοσίας γίνεται σύμφωνα με το παράρτημα. Παράδειγμα: Υπολογισμός Μισθολογίου (Προσοχή, υπάρχουν αποκλίσεις από τον τρόπο που θα υπολογίζετε εσείς την μισθοδοσία).

Α. Κώδικας

```
1.  public void CalculateSalary(Employee employee, ref double grossSalary, ref double netIncome)
    {
2.
3.      string categoryEmployee = employee.category;
4.      string studiesEmployee = employee.studies;
5.      int workExperienceEmployee = employee.workExperience;
6.      int childrenEmployee = employee.children;
7.
8.      double minimumWagePE = 1092;
9.      double minimumWageTE = 1037;
10.     double minimumWage = 0;
11.
12.     int j = 2;
13.     int i = 0;
14.
```

```
15.
16.     // 1. Υπολογισμός βασικού μισθού
17.     for (i = 0; i < 38; i += 2) {
18.         if (workExperienceEmployee >= i && workExperienceEmployee < j) {
19.             // Έλεγχος αν είναι ΡΕ
20.             if (categoryEmployee.Equals("PE") || categoryEmployee.Equals("ΠΕ")) {
21.                 minimumWage = minimumWagePE;
22.             } // Έλεγχος αν είναι ΤΕ
23.             else if (categoryEmployee.Equals("TE") || categoryEmployee.Equals("ΤΕ")) {
24.                 minimumWage = minimumWageTE;
25.             }
26.             break;
27.         } else {
28.             j = j + 2;
29.
30.             // Έλεγχος αν είναι ΡΕ
31.             if (categoryEmployee.Equals("PE") || categoryEmployee.Equals("ΠΕ")) {
32.                 minimumWagePE += 59;
33.             } // Έλεγχος αν είναι ΤΕ
34.             else if (categoryEmployee.Equals("TE") || categoryEmployee.Equals("ΤΕ")) {
35.                 minimumWageTE += 55;
36.
37.             }
38.         }
39.     }
40.
41.     // Έλεγχος πτυχίου που έχει, αλλιώς ξέρουμε by default ότι είναι χωρίς μεταπτυχιακό
42.     if (studiesEmployee.Equals("Μεταπτυχιακό") || studiesEmployee.Equals("MSc")) {
43.         // Έλεγχος αν είναι ΡΕ
44.         if (categoryEmployee.Equals("PE") || categoryEmployee.Equals("ΠΕ")) {
45.             minimumWage += 2 * 59;
46.         } // Έλεγχος αν είναι ΤΕ
47.         else if (categoryEmployee.Equals("TE") || categoryEmployee.Equals("ΤΕ")) {
48.             minimumWage += 2 * 55;
49.         }
50.     } else if (studiesEmployee.Equals("Διδακτορικό") || studiesEmployee.Equals("PhD")) {
51.         // Έλεγχος αν είναι ΡΕ
52.         if (categoryEmployee.Equals("PE") || categoryEmployee.Equals("ΠΕ")) {
53.             minimumWage += 6 * 59;
54.         } // Έλεγχος αν είναι ΤΕ
55.         else if (categoryEmployee.Equals("TE") || categoryEmployee.Equals("ΤΕ")) {
56.             minimumWage += 6 * 55;
57.         }
58.     } else if (studiesEmployee.Equals("Χωρίς Μεταπτυχιακό") ||
    studiesEmployee.Equals("NoMSc")) {
59.
60.     }
61.     //Console.WriteLine(minimumWage);
62.
63.     // 2. Υπολογισμός Οικογενειακής παροχής και  Μικτές αποδοχές
64.     if (childrenEmployee != 0) {
65.         switch (childrenEmployee) {
66.             case 1:
67.                 grossSalary = minimumWage + 50;
68.                 break;
69.             case 2:
70.                 grossSalary = minimumWage + 70;
71.                 break;
72.             case 3:
73.                 grossSalary = minimumWage + 120;
74.                 break;
75.             case 4:
76.                 grossSalary = minimumWage + 170;
77.                 break;
78.             case 5:
```

```
                grossSalary = minimumWage + 240;
                break;
            default:
                grossSalary = minimumWage + 310;
                break;
        }
    } else
        grossSalary = minimumWage;
    //Console.WriteLine(grossSalary);

    // 3. Υπολογισμός κρατήσεων
    double holdIka = grossSalary * 0.16;
    double holdOAED = grossSalary * 0.01;

    double hold = holdIka + holdOAED;
    //Console.WriteLine(hold);

    double katharesApodoxes = grossSalary - hold;
    double yearlyKatharesApodoxes = 12 * katharesApodoxes;
    //Console.WriteLine(yearlyKatharesApodoxes);

      double yearlyTax = 0;

      // 4. Υπολογισμός Φόρου
      if (yearlyKatharesApodoxes <= 10000) {
          yearlyTax = 0.09 * yearlyKatharesApodoxes;
      } else if (yearlyKatharesApodoxes <= 20000) {
          yearlyTax = 0.22 * yearlyKatharesApodoxes;
      } else if (yearlyKatharesApodoxes <= 30000) {
          yearlyTax = 0.28 * yearlyKatharesApodoxes;
      } else if (yearlyKatharesApodoxes <= 40000) {
          yearlyTax = 0.36 * yearlyKatharesApodoxes;
      } else if (yearlyKatharesApodoxes > 40000) {
          yearlyTax = 0.44 * yearlyKatharesApodoxes;
      } else {
          // print error or something
      }
      //Console.WriteLine(yearlyTax);


      int childlrenEmployeeTemp = 0;
      double yearlyFinalTax = 0;
      // αν είναι αρνητικός ο αριθμός παιδιών ;;

      // 4.1 Υπολογισμός Φόρου έκπτωσης
      switch (childrenEmployee) {
          case 0:
              yearlyFinalTax = yearlyTax - 777;
              break;
          case 1:
              yearlyFinalTax = yearlyTax - 810;
              break;
          case 2:
              yearlyFinalTax = yearlyTax - 900;
              break;
          case 3:
              yearlyFinalTax = yearlyTax - 1120;
              break;
          case 4:
              yearlyFinalTax = yearlyTax - 1340;
              break;
          default:
              childlrenEmployeeTemp = childrenEmployee - 4;
              yearlyFinalTax = yearlyTax - (1340 + 220 * childlrenEmployeeTemp);
              break;
```

```
144.      }
145.      //Console.WriteLine(yearlyFinalTax);
146.
147.      // Φόρος το μήνα
148.      double taxPerMonthFinal = yearlyFinalTax / 12;
149.      // Console.WriteLine(taxPerMonthFinal);
150.
151.      // Καθαρό εισόδημα = μικτές αποδοχές - κρατήσεις - φόρος
152.      netIncome = grossSalary - hold - taxPerMonthFinal;
153.      //Console.WriteLine(netIncome);
154.  }
```

Β. Πίνακας – περιπτώσεις ελέγχου

Περ.Ελ.	**Δοκιμαστικά Δεδ. Εισόδου**				**Αναμ. Αποτ.**	
#id	**employee**				**grossSalary**	**netIncome**
	category	studies	workExperience	children		
1	ΠΕ	PhD	4	2	true	Alpha Bank
2	ΠΕ	Χωρίς Μεταπτυχιακό	2	1	false	""
3	ΠΕ	PhD	5	1	false	""
4	ΠΕ	Διδακτορικό	4	2	1634.00	1132.8516
5	ΤΕ	PhD	2	3	1542.00	1091.6241333333
6	ΤΕ	Μεταπτυχιακό	4	2	1327.00	934.0998
7	ΤΕ	PhD	1	0	1367.00	949.75
8	ΠΕ	MSc	3	4	1439.00	1043.27526666667
9	ΠΕ	MSc	7	1	1437.00	997.8138
10	ΤΕ	Χωρίς Μεταπτυχιακό	27	2	1822.00	1254.5628
11	ΤΕ	Χωρίς Μεταπτυχιακό	34	2	2042.00	1295.2992
12	ΠΕ	Μεταπτυχιακό	28	2	2106.00	1333.5456

Γ. Unit Tests

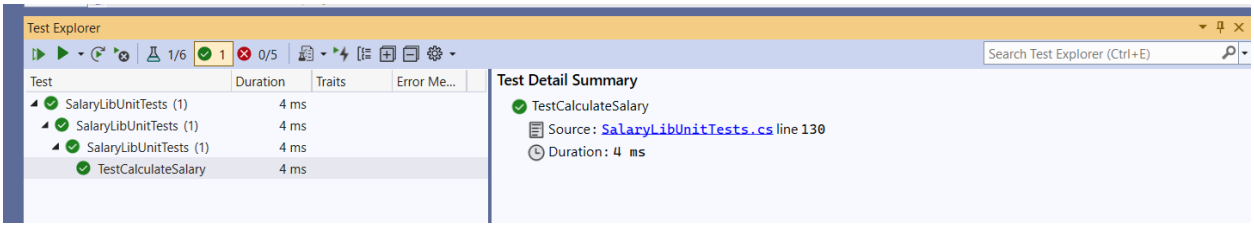
```
1.  public void TestCalculateSalary()
2.          {
3.              //  CalculateSalary(Employee employee, ref double grossSalary, ref double
    netIncome)
4.
5.              // Employee  category, studies, workExperience, children
6.
7.              //Δημιουργία ενός αντικειμένου της κλάσης του dll που θέλουμε τα τεστάρουμε
8.              SalaryLib.SalaryLib salaryLib = new SalaryLib.SalaryLib();
9.
```

```
10.             //Δημιουργία Περιπτώσεων Ελέγχου
11.             object[,] testcases =
12.             {
13.                 {1, "PE", "PhD", 4, 2, 1634.00,1132.8516},
14.                 {2, "PE","Χωρίς Μεταπτυχιακό",2,1,1201.00,845.0274},
15.                 {3, "ΠΕ", "PhD", 5, 1,1614.00,1112.4036},
16.                 {4, "ΠΕ", "Διδακτορικό", 4, 2,1634.00,1132.8516},
17.                 {5, "ΤΕ", "PhD", 2, 3, 1542.00,1091.62413333333},
18.                 {6, "ΤΕ", "Μεταπτυχιακό", 4, 2, 1327.00,934.0998},
19.                 {7, "ΤΕ", "PhD", 1, 0, 1367.00,949.75},
20.                 {8, "ΠΕ", "MSc", 3, 4, 1439.00,1043.27526666667},
21.                 {9, "PE", "MSc", 7, 1, 1437.00,997.8138},
22.                 {10, "ΤΕ", "Χωρίς Μεταπτυχιακό", 27, 2, 1822.00,1254.5628},
23.                 {11, "ΤΕ", "Χωρίς Μεταπτυχιακό", 34, 2,2042.00,1295.2992},
24.                 {12, "PE", "Μεταπτυχιακό", 28, 2, 2106.00,1333.5456},
25.
26.                 //Προσθήκη όλων των περιπτώσεων ελέγχου που αφορούν τη συγκεκριμένη
    testmethod
27.             };
28.
29.
30.             //Αρχικοποίηση δείκτη περιπτώσεων ελέγχου
31.             int i = 0;
32.             bool failed = false;
33.
34.             //Προσπέλαση και εκτέλεση περιπτώσεων ελέγχου
35.             for (i = 0; i < testcases.GetLength(0); i++)
36.             //Για κάθε περίπτωση ελέγχου, δηλαδή για κάθε γραμμή i του πίνακα testcases
37.             {
38.                 try
39.                 {
40.                     //Δήλωση ref μεταβλητών μεθόδου,
41.                     //προκειμένου να αποθηκευτούν οι τιμές που θα επιστρέψει η μέθοδος
42.                     double grossSalary = 0;
43.                     double netIncome = 0;
44.
45.                     // Δημιουργία employee
46.                     SalaryLib.SalaryLib.Employee employee = new
    SalaryLib.SalaryLib.Employee();
47.
48.                     // Αρχικοποίηση αντικειμένου employee
49.                     employee.category = (string)testcases[i, 1];
50.                     employee.studies = (string)testcases[i, 2];
51.                     employee.workExperience = (int)testcases[i, 3];
52.                     employee.children = (int)testcases[i, 4];
53.
54.
55.                     //Καλούμε την μέθοδο που θέλουμε να εξετάσουμε δίνοντας ως παραμέτρους
    τα
56.                     //στοιχεία της περίπτωσης ελέγχου,
57.                     //δηλαδή τα αντίστοιχα στοιχεία της γραμμής i του πίνακα testcases,
58.                     //και ως ref τις μεταβλητές που δηλώσαμε
59.                     salaryLib.CalculateSalary(employee, ref grossSalary, ref netIncome);
60.
61.                     //Καλούμε την Assert.AreEqual δίνοντας ως παραμέτρους τις τιμές
62.                     //που θέλουμε να συγκρίνουμε
63.
64.                     Assert.AreEqual(Math.Round((double)testcases[i, 5], 2),
    Math.Round(grossSalary,2));
65.                     Assert.AreEqual(Math.Round((double)testcases[i, 6],2),
    Math.Round(netIncome,2));
66.
67.                 }
68.                 catch (Exception e)
69.                 {
```

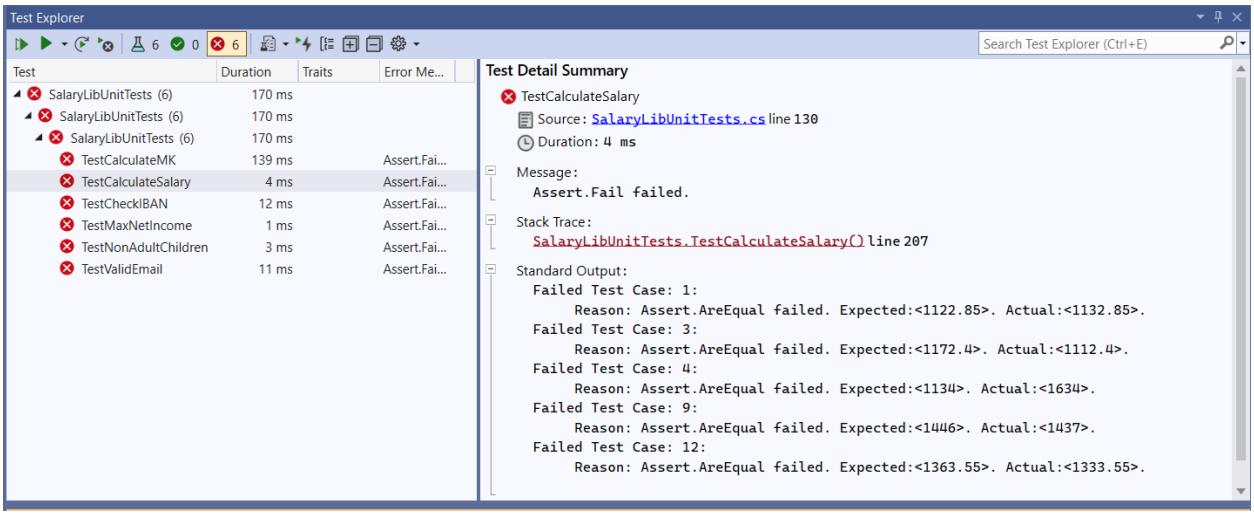
```
70.                     //Απέτυχε η περίπτωση ελέγχου
71.                     failed = true;
72.                     //Καταγράφουμε την περίπτωση ελέγχου που απέτυχε
73.                     Console.WriteLine("Failed Test Case: {0}: \n \t Reason: {1} ",
    (int)testcases[i, 0], e.Message);
74.                 };
75.             };
76.
77.             //Στην περίπτωση που κάποια περίπτωση ελέγχου απέτυχε, πέταξε εξαίρεση.
78.             if (failed) Assert.Fail();
79.         }
```

Δ. Αναφορές Ελέγχου

Με σωστά δεδομένα



Αναφορές σφαλμάτων



4. calculateMK(string hiringDate, string studies, ref int MK, ref int excessYears, ref int excessMonths, ref int excessDays)

Η μέθοδος αυτή δέχεται ως παράμετρο την ημερομηνία πρόσληψης ενός υπαλλήλου και το επίπεδο μεταπτυχιακών σπουδών του (Χωρίς Μεταπτυχιακό/NoMSc, Μεταπτυχιακό/MSc, Διδακτορικό/PhD), και επιστρέφει το μισθολογικό κλιμάκιο (ΜΚ) που ανήκει, καθώς και τον πλεονάζοντα χρόνο της προϋπηρεσίας του στο μισθολογικό κλιμάκιο που έχουν ενταχθεί (π.χ. 1 χρόνος, 2 μήνες και 18 μέρες). Τα αποτελέσματα αυτά επιστρέφονται μέσω των αντίστοιχων by ref παραμέτρων που δέχεται. Ο υπολογισμός του ΜΚ γίνεται σύμφωνα με το παράρτημα.

Α. Κώδικας

```
1.  public void CalculateMK(string hiringDate, string studies, ref int MK, ref int ExcessYears,
    ref int ExcessMonths, ref int ExcessDays) {
2.      /**************Υπολογισμός χρόνος προϋπηρέσίας με ημερομηνία έως 2022/02/25
    **************/
3.
4.      DateTime dob = Convert.ToDateTime(hiringDate);
5.      // Method to Calculate age from given date in C#
6.      DateTime Today = Convert.ToDateTime("2022/02/25");
7.      int Years = new DateTime(Today.Subtract(dob).Ticks).Year - 1;
8.      DateTime PastYearDate = dob.AddYears(Years);
9.      int Months = 0;
10.     int i = 0;
11.
12.     for (i = 1; i <= 12; i++) {
13.         if (PastYearDate.AddMonths(i) == Today) {
14.             Months = i;
15.             break;
16.         } else if (PastYearDate.AddMonths(i) >= Today) {
17.             Months = i - 1;
18.             break;
19.         }
20.     }
21.
22.     int Days = Today.Subtract(PastYearDate.AddMonths(Months)).Days;
23.     int Hours = Today.Subtract(PastYearDate).Hours;
24.     int Minutes = Today.Subtract(PastYearDate).Minutes;
25.     int Seconds = Today.Subtract(PastYearDate).Seconds;
26.
27.
28.     int j = 2;
29.     MK = 1;
30.
31.     /**************Υπολογισμός σε ποιο MK ανήκει**************/
32.     for (i = 0; i < 38; i += 2) {
33.         if (Years >= i && Years < j) {
34.             string studiesEmployee = studies;
35.             // Έλεγχος πτυχίου που έχει, αλλιώς ξέρουμε by default ότι είναι χωρίς
    μεταπτυχιακό
36.             if (studiesEmployee.Equals("Μεταπτυχιακό") || studiesEmployee.Equals("MSc")) {
37.                 MK += 2;
38.             } else if (studiesEmployee.Equals("Διδακτορικό") ||
    studiesEmployee.Equals("PhD")) {
39.                 MK += 6;
40.             } else if (studiesEmployee.Equals("Χωρίς Μεταπτυχιακό") ||
    studiesEmployee.Equals("NoMSc")) {
41.                 MK = MK;
42.             }
43.             break;
44.         } else {
45.             j = j + 2;
46.             MK += 1;
47.         }
48.     }
49.
50.     /**************Υπολογισμός πόσο χρόνο ανήκει στο συγκεκριμένο MK**************/
51.     ExcessYears = Years % 2;
52.     ExcessMonths = Months;
53.     ExcessDays = Days;
54.
55. }
```

Β. Πίνακας – περιπτώσεις ελέγχου

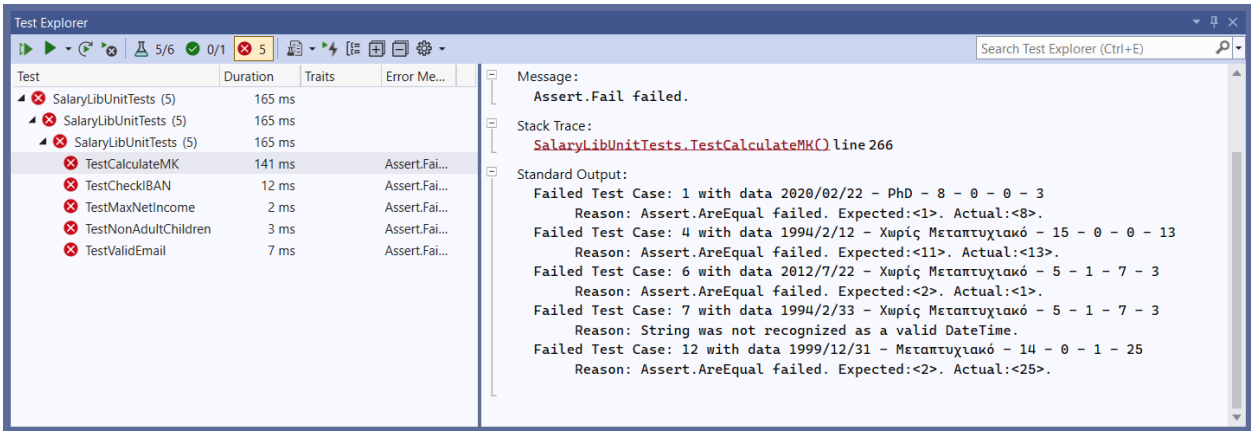
Περ.Ελ.	**Δοκιμαστικά Δεδ. Εισόδου**		**Αναμ. Αποτ.**			
#id	**hiringDate**	**studies**	**MK**	**ExcessYears**	**ExcessMonths**	**ExcessDays**
1	2020/02/22	PhD	8	0	0	3
2	2014/05/7	Μεταπτυχιακό	6	1	9	18
3	2012/9/23	Διδακτορικό	11	1	5	2
4	1994/2/12	Χωρίς Μεταπτυχιακό	15	0	0	13
5	2019/10/23	PhD	8	0	4	2
6	2012/7/22	Χωρίς Μεταπτυχιακό	5	1	7	3
7	1994/2/33	Χωρίς Μεταπτυχιακό	Error: wrong date			
8	2005/9/5	MSc	11	0	5	20
9	2013/8/8	PhD	11	06	17	
10	2017/3/9	PhD	9	0	11	16
11	1995/1/1	Μεταπτυχιακό	16	1	1	24
12	1999/12/31	Μεταπτυχιακό	14	0	1	2

Γ. Unit Tests

```
1.          [TestMethod]
2.          public void TestCalculateMK()
3.          {
4.              //Δημιουργία ενός αντικειμένου της κλάσης του dll που θέλουμε τα τεστάρουμε
5.              SalaryLib.SalaryLib salaryLib = new SalaryLib.SalaryLib();
6.
7.              //Δημιουργία Περιπτώσεων Ελέγχου
8.              object[,] testcases =
9.              {
10.                 {1,"2020/02/22","PhD", 8, 0, 0, 3},
11.                 {2,"2014/05/7","Μεταπτυχιακό", 6, 1, 9, 18},
12.                 {3,"2012/9/23","Διδακτορικό", 11,1,5,2},
13.                 {4,"1994/2/12","Χωρίς Μεταπτυχιακό", 15,0,0,13},
14.                 {5,"2019/10/23","PhD",8,0,4,2 },
15.                 {6,"2012/7/22","Χωρίς Μεταπτυχιακό", 5,1,7,3},
16.                 {7,"1994/2/33","Χωρίς Μεταπτυχιακό", 11,3,4,5},
17.                 {8,"2005/9/5","MSc", 11,0,5,20},
18.                 {9,"2013/8/8","PhD", 11,0,6,17},
19.                 {10,"2017/3/9","PhD",9,0,11,16},
20.                 {11,"1995/1/1","Μεταπτυχιακό",16,1,1,24},
21.                 {12,"1999/12/31","Μεταπτυχιακό",14,0,1,2}
22.             };
23.
24.             //Αρχικοποίηση δείκτη περιπτώσεων ελέγχου
25.             int i = 0;
26.             bool failed = false;
27.             int MK = 0;
28.             int ExcessYears = 0;
29.             int ExcessMonths = 0;
30.             int ExcessDays = 0;
```

```
31.             //Προσπέλαση και εκτέλεση περιπτώσεων ελέγχου
32.             for (i = 0; i < testcases.GetLength(0); i++)
33.             //Για κάθε περίπτωση ελέγχου, δηλαδή για κάθε γραμμή i του πίνακα testcases
34.             {
35.                 try
36.                 {
37.
38.
39.                     salaryLib.CalculateMK((string)testcases[i, 1], (string)testcases[i, 2],
    ref MK, ref ExcessYears, ref ExcessMonths, ref ExcessDays);
40.
41.                     Assert.AreEqual((int)testcases[i, 3], MK);
42.                     Assert.AreEqual((int)testcases[i, 4], ExcessYears);
43.                     Assert.AreEqual((int)testcases[i, 5], ExcessMonths);
44.                     Assert.AreEqual((int)testcases[i, 6], ExcessDays);
45.
46.                 }
47.                 catch (Exception e)
48.                 {
49.                     failed = true;
50.                     Console.WriteLine("Failed Test Case: {0} with data {1} - {2} - {3} - {4}
    - {5} - {6}\n \t Reason: {7} ", (int)testcases[i, 0], (string)testcases[i, 1],
    (string)testcases[i, 2], MK, ExcessYears,ExcessMonths, ExcessDays, e.Message);
51.                 };
52.             };
53.
54.
55.             //Στην περίπτωση που κάποια περίπτωση ελέγχου απέτυχε, πέταξε εξαίρεση.
56.             if (failed) Assert.Fail();
57.         }
```

Δ. Αναφορές Ελέγχου



5. int nonAdultChildren(string[] childrenBirthday)

Η συνάρτηση αυτή υπολογίζει και επιστρέφει το πλήθος των ανήλικων παιδιών που υπάρχουν στον πίνακα που δέχεται ως παράμετρο. Ο πίνακας αυτός περιλαμβάνει τις ημερομηνίες γέννησης των παιδιών ενός υπαλλήλου.

Α. Κώδικας

```
1.  public int NonAdultChildren(string[] ChildrenBirthday) {
2.
3.      int numberOfNonAdults = 0;
4.      string dateOfBirth = null;
5.      int age = 0;
6.
7.      // check each entry of the array
8.      foreach(string entry in ChildrenBirthday) {
9.          dateOfBirth = entry;
10.         DateTime dob = Convert.ToDateTime(dateOfBirth);
11.         age = CalculateAge(dob);
12.         if (age < 18) {
13.             numberOfNonAdults++;
14.         }
15.     }
16.     return numberOfNonAdults;
17. }
```

Β. Πίνακας – περιπτώσεις ελέγχου

Περ.Ελ.	**Δοκιμαστικά Δεδ. Εισόδου**	**Αναμ. Αποτ.**
#id	**string[] ChildrenBirthday**	**Αριθμός ανηλίκων**

1	string[] dobChildren01 = new string[] {
	"1991/10/28",
	"2020/12/01",
	"2015/3/2"
	};
		2

2	string[] dobChildren02 = new string[] {
	"1991/10/28",
	"2020/12/01",
	"1998/3/2"
	};
		1

3	string[] dobChildren03 = new string[]
	{
	"2020/01/20",
	"1995/08/18",
	"1993/10/4"
	};
		1

4	string[] dobChildren04 = new string[]
	{
	"2015/02/13"
	};
		1

5	string[] dobChildren05 = new string[]
	{
	"2019/02/18"
	};
		1

6	string[] dobChildren06 = new string[] { "2020/04/15",	1

	"1996/9/12" };	

7	string[] dobChildren07 = new string[]
	{
	"2003/03/7",
	"1990/7/3"
	};
		0

8	string[] dobChildren08 = new string[]
	{
	"2009/01/18",
	"1992/4/31",
	"2020/5/12"
	};
		Error, no valid date

9	string[] dobChildren09 = new string[]
	{
	"2000/06/23"
	};
		0

10	string[] dobChildren10 = new string[]
	{
	"2019/03/14",
	"1985/3/5"
	};
		1

11	string[] dobChildren11 = new string[]
	{
	"2012/6/7",
	"1999/12/3",
	"2019/03/14",
	"1980/2/5"
	};
		2

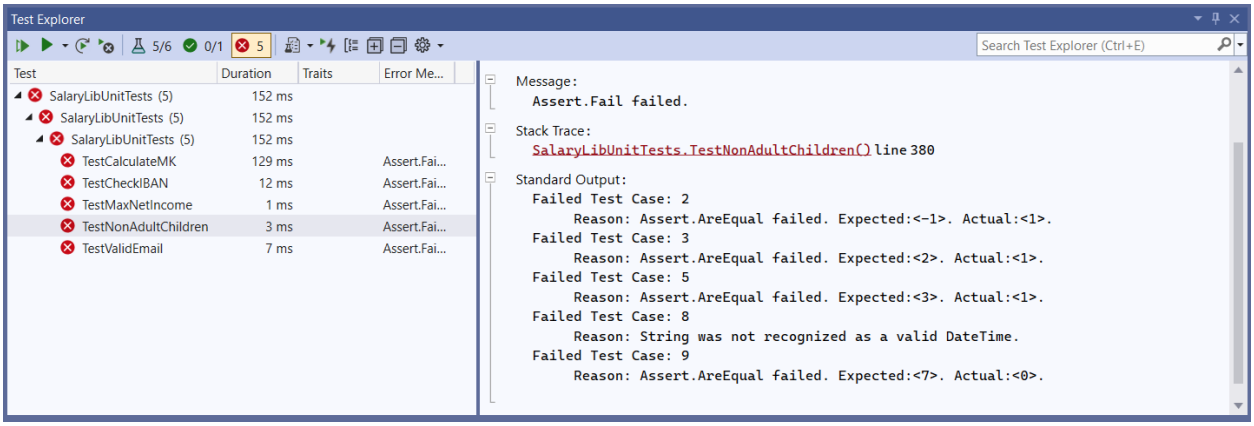
Γ. Unit Tests

```
1.  [TestMethod]
2.          public void TestNonAdultChildren()
3.          {
4.              // Δημιουργία πινάκων με ημερομηνίες παιδιών κάθε υπαλλήλου
5.
6.              // πίνακας παιδιών υπάλληλου01
7.              string[] dobChildren01 = new string[] {
8.              "1991/10/28",
9.              "2020/12/01",
10.             "2015/3/2"
11.             };
12.
13.             string[] dobChildren02 = new string[] {
14.             "1991/10/28",
15.             "2020/12/01",
16.             "1998/3/2"
17.             };
18.
19.             string[] dobChildren03 = new string[]
20.             {
```

```
                "2020/01/20",
                "1995/08/18",
                "1993/10/4"
            };

            string[] dobChildren04 = new string[]
            {
                "2015/02/13"
            };

            string[] dobChildren05 = new string[]
            {
                "2019/02/18"
            };

            string[] dobChildren06 = new string[]
            {
                "2020/04/15",
                "1996/9/12"
            };
            string[] dobChildren07 = new string[]
            {
                "2003/03/7",
                "1990/7/3"
            };
            string[] dobChildren08 = new string[]
            {
                "2009/01/18",
                "1992/4/31",
                "2020/5/12"
            };
            string[] dobChildren09 = new string[]
            {
                "2000/06/23"
            };
            string[] dobChildren10 = new string[]
            {
                "2019/03/14",
                "1985/3/5"
            };
            string[] dobChildren11 = new string[]
            {
                "2012/6/7",
                "1999/12/3",
                "2019/03/14",
                "1980/2/5"
            };
            //Δημιουργία ενός αντικειμένου της κλάσης του dll που θέλουμε τα τεστάρουμε
            SalaryLib.SalaryLib salaryLib = new SalaryLib.SalaryLib();

            //Δημιουργία Περιπτώσεων Ελέγχου
            object[,] testcases =
            {
                { 1, dobChildren01, 2},
                { 2, dobChildren02, 1},
                { 3, dobChildren03, 1},
                { 4, dobChildren04, 1},
                { 5, dobChildren05, 1},
                { 6, dobChildren06, 1},
                { 7, dobChildren07, 0},
                { 8, dobChildren08, 2},
                { 9, dobChildren09, 0},
                { 10, dobChildren10, 1},
                { 10, dobChildren11, 2}
            };
```

```
86.
87.             //Αρχικοποίηση δείκτη περιπτώσεων ελέγχου
88.             int i = 0;
89.             bool failed = false;
90.
91.             //Προσπέλαση και εκτέλεση περιπτώσεων ελέγχου
92.             for (i = 0; i < testcases.GetLength(0); i++)
93.             //Για κάθε περίπτωση ελέγχου, δηλαδή για κάθε γραμμή i του πίνακα testcases
94.             {
95.                 try
96.                 {
97.                     //Καλούμε την Assert.AreEqual δίνοντας ως παραμέτρους τα στοιχεία της
    περίπτωσης ελέγχου,
98.                     //δηλαδή τα αντίστοιχα στοιχεία της γραμμής i του πίνακα testcases
99.                     Assert.AreEqual((int)testcases[i, 2],
    salaryLib.NonAdultChildren((string[])testcases[i, 1]));
100.                }
101.                catch (Exception e)
102.                {
103.                    //Απέτυχε η περίπτωση ελέγχου
104.                    failed = true;
105.                    //Καταγράφουμε την περίπτωση ελέγχου που απέτυχε
106.                    Console.WriteLine("Failed Test Case: {0} \n \t Reason: {1}
    ",(int)testcases[i, 0], e.Message);
107.                };
108.            };
109.
110.            //Στην περίπτωση που κάποια περίπτωση ελέγχου απέτυχε, πέταξε εξαίρεση.
111.            if (failed) Assert.Fail();
112.        }
```

Δ. Αναφορές Ελέγχου



6. double maxNetIncome(Employee[] empls)

Η συνάρτηση αυτή επιστρέφει τον μεγαλύτερο καθαρό μισθό ενός πλήθους υπαλλήλων. Η συγκεκριμένη συνάρτηση κάνει κλήση της συνάρτησης CalculateSalary.

Α. Κώδικας

```
1.  public double MaxNetIncome(Employee[] Empls) {
2.      double maxNetIncome = 0.0;
```

```
3.      double grossSalary = 0;
4.      double netIncome = 0;
5.
6.      foreach(Employee emp in Empls) {
7.          // Κλήση της συνάρτησης CalculateSalary για να ανακτήσουμε το netIncome του κάθε
    employee
8.          CalculateSalary(emp, ref grossSalary, ref netIncome);
9.
10.         // Υπολογισμός maxNetIncome
11.         if (maxNetIncome < netIncome) {
12.             maxNetIncome = netIncome;
13.         }
14.     }
15.     return maxNetIncome;
16. }
```

Β. Πίνακας – περιπτώσεις ελέγχου

Περ.Ελ.	**Δοκιμαστικά Δεδ. Εισόδου**	**Αναμ. Αποτ.**
#id	**Employee[] Empls**	**Μέγιστον netIncome**
1	Employee[] arrayEmployee1 = new Employee[] { new Employee("PE", "PhD", 4, 2), // 1132.8516 new Employee("ΠΕ", "Χωρίς Μεταπτυχιακό", 2, 1) //845.0274 };	1132.8516
2	Employee[] arrayEmployee2 = new Employee[] { new Employee("PE", "PhD", 5, 1), // 1112.4036 new Employee("ΠΕ", "Διδακτορικό", 4, 2) //1132.8516 };	1132.8516
3	Employee[] arrayEmployee3 = new Employee[] { new Employee("ΤΕ", "PhD", 2, 3), //1091.62413333333 new Employee("ΤΕ", "Μεταπτυχιακό", 4, 2) // 934.0998 };	1091.62413333333
4	Employee[] arrayEmployee4 = new Employee[] { new Employee("ΤΕ", "PhD", 1, 0), //949.75 new Employee("ΠΕ", "MSc", 3, 4) // 1043.27526666667 };	1043.27526666667
5	Employee[] arrayEmployee5 = new Employee[]	1254.5628

	{
	new Employee("PE", "MSc", 3, 2), //997.8138
	new Employee("TE", "Χωρίς Μεταπτυχιακό", 27, 2) // 1254.5628
	};	

6	Employee[] arrayEmployee6 = new Employee[]
	{
	new Employee("TE", "Χωρίς Μεταπτυχιακό", 34, 2), // 1295.2992
	new Employee("ΠΕ", "Μεταπτυχιακό", 28, 2) // 1333.5456
	};
		1333.5456

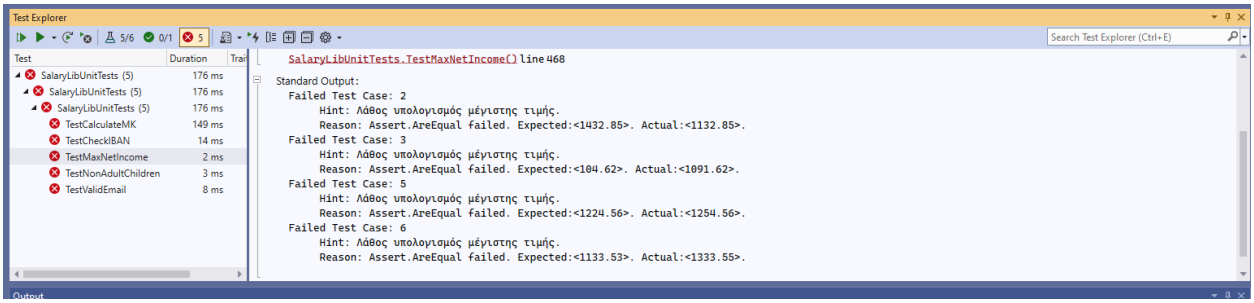
Γ. Unit Tests

```
1.  [TestMethod]
2.          public void TestMaxNetIncome()
3.          {
4.
5.              const string msg1 = "Λάθος υπολογισμός μέγιστης τιμής.";
6.
7.              //Δημιουργία ενός αντικειμένου της κλάσης του dll που θέλουμε τα τεστάρουμε
8.              SalaryLib.SalaryLib salaryLib = new SalaryLib.SalaryLib();
9.
10.
11.             // Δημιουργία δεδομένων για τις περιπτώσεις ελέγχου
12.             Employee[] arrayEmployee1 = new Employee[]
13.             {
14.                 new Employee("PE", "PhD", 4, 2),   // 1132.8516
15.                 new Employee("ΠΕ", "Χωρίς Μεταπτυχιακό", 2, 1) //845.0274
16.             };
17.
18.             Employee[] arrayEmployee2 = new Employee[]
19.             {
20.                 new Employee("PE", "PhD", 5, 1),   // 1112.4036
21.                 new Employee("ΠΕ", "Διδακτορικό", 4, 2) //1132.8516
22.             };
23.
24.             Employee[] arrayEmployee3 = new Employee[]
25.             {
26.                 new Employee("TE", "PhD", 2, 3), //1091.62413333333
27.                 new Employee("TE", "Μεταπτυχιακό", 4, 2) //  934.0998
28.             };
29.
30.             Employee[] arrayEmployee4 = new Employee[]
31.             {
32.                 new Employee("TE", "PhD", 1, 0), //949.75
33.                 new Employee("ΠΕ", "MSc", 3, 4) // 1043.27526666667
34.             };
35.
36.             Employee[] arrayEmployee5 = new Employee[]
37.             {
38.                 new Employee("PE", "MSc", 3, 2), //997.8138
39.                 new Employee("TE", "Χωρίς Μεταπτυχιακό", 27, 2) // 1254.5628
40.             };
41.
42.             Employee[] arrayEmployee6 = new Employee[]
```

```
43.             {
44.                  new Employee("ΤΕ", "Χωρίς Μεταπτυχιακό", 34, 2), // 1295.2992
45.                  new Employee("ΠΕ", "Μεταπτυχιακό", 28, 2) // 1333.5456
46.             };
47.
48.             //Δημιουργία Περιπτώσεων Ελέγχου
49.             object[,] testcases =
50.             {
51.                 {1, arrayEmployee1, 1132.8516, msg1},
52.                 {2, arrayEmployee2, 1132.8516, msg1},
53.                 {3, arrayEmployee3, 1091.62413333333, msg1},
54.                 {4, arrayEmployee4, 1043.28,msg1 },
55.                 {5, arrayEmployee5, 1254.5628 , msg1},
56.                 {6, arrayEmployee6, 1333.5456, msg1 }
57.                 //Προσθήκη όλων των περιπτώσεων ελέγχου που αφορούν τη συγκεκριμένη
    testmethod
58.             };
59.
60.             //Αρχικοποίηση δείκτη περιπτώσεων ελέγχου
61.             int i = 0;
62.             bool failed = false;
63.
64.             //Προσπέλαση και εκτέλεση περιπτώσεων ελέγχου
65.             for (i = 0; i < testcases.GetLength(0); i++)
66.             //Για κάθε περίπτωση ελέγχου, δηλαδή για κάθε γραμμή i του πίνακα testcases
67.             {
68.                 try
69.                 {
70.                     //Καλούμε την Assert.AreEqual δίνοντας ως παραμέτρους τα στοιχεία της
    περίπτωσης ελέγχου,
71.                     //δηλαδή τα αντίστοιχα στοιχεία της γραμμής i του πίνακα testcases
72.                     Assert.AreEqual(Math.Round((double)testcases[i, 2], 2),
    Math.Round(salaryLib.MaxNetIncome((Employee[])testcases[i, 1]), 2));
73.
74.                 }
75.                 catch (Exception e)
76.                 {
77.                     //Απέτυχε η περίπτωση ελέγχου
78.                     failed = true;
79.                     //Καταγράφουμε την περίπτωση ελέγχου που απέτυχε
80.                     Console.WriteLine("Failed Test Case: {0} \n \t Hint: {1} \n \t Reason:
    {2} ", (int)testcases[i, 0], (string)testcases[i, 3], e.Message);
81.                 };
82.             };
83.
84.             //Στην περίπτωση που κάποια περίπτωση ελέγχου απέτυχε, πέταξε εξαίρεση.
85.             if (failed) Assert.Fail();
86.         }
87.
```

Δ. Αναφορές Ελέγχου

Αναφορές Σφαλμάτων



7. Βοηθητική συνάρτηση για τον υπολογισμό ηλικίας

```
1.  public int CalculateAge(DateTime p_Dob) {
2.      // Method to Calculate age from Date of Birth in C#
3.      DateTime Today = DateTime.Now;
4.      int Years = new DateTime(DateTime.Now.Subtract(p_Dob).Ticks).Year - 1;
5.      DateTime PastYearDate = p_Dob.AddYears(Years);
6.      int Months = 0;
7.      for (int i = 1; i <= 12; i++) {
8.          if (PastYearDate.AddMonths(i) == Today) {
9.              Months = i;
10.             break;
11.         } else if (PastYearDate.AddMonths(i) >= Today) {
12.             Months = i - 1;
13.             break;
14.         }
15.     }
16.     int Days = Today.Subtract(PastYearDate.AddMonths(Months)).Days;
17.     int Hours = Today.Subtract(PastYearDate).Hours;
18.     int Minutes = Today.Subtract(PastYearDate).Minutes;
19.     int Seconds = Today.Subtract(PastYearDate).Seconds;
20.
21.     return Years;
22. }
```

ΔΟΜΗ ΔΕΔΟΜΕΝΩΝ

Employee με πεδία

- string category (κατηγορία του υπαλλήλου – τιμές: ΠΕ/PE ή ΤΕ/TE),
- string studies (επίπεδο μεταπτυχιακών σπουδών – τιμές: Χωρίς Μεταπτυχιακό/NoMSc, Μεταπτυχιακό/MSc, Διδακτορικό/PhD),
- int WorkExperience (έτη προϋπηρεσίας – τιμές: από 0 έως 38)
- int Children (πλήθος των ανήλικων παιδιών – τιμές: από 0 έως 6)

Α. Κώδικας

```
1.  public struct Employee {
2.
3.      // Δήλωση μεταβλητών μέλη
4.      public string category {
5.          get;
6.          set;
7.      } // τιμές ΠΕ/PE ή ΤΕ/TE
8.      public string studies; // τιμές Χωρίς Μεταπτυχιακό/NoMSc, Μεταπτυχιακό/MSc,
    Διδακτορικό/PhD
9.      public int workExperience; // τιμές 0<=workExperience<=38
10.     public int children; // τιμές 0<=children<=6
11.
12.     // Δήλωση constructor
13.     public Employee(string category, string studies, int workExperience, int children) {
14.         this.category = category;
15.         this.studies = studies;
16.         this.workExperience = workExperience;
17.         this.children = children;
18.     }
19. }
```

ΣΥΜΠΕΡΑΣΜΑ

Έχοντας πλέον εκπονήσει την εργασία, καταλάβαμε τη σημασία των unit tests και πως μπορούμε αυτά να τα υλοποιήσουμε. Επιπλέον, μάθαμε εισαγωγικές έννοιες της γλώσσας προγραμματισμού της C#, η οποία είναι παρόμοια με τη γνωστή αντικειμενοστρεφή γλώσσα, JAVA. Εμβαθύναμε τις γνώσεις μας για τις περιπτώσεις ελέγχου και πως μπορούμε να σχεδιάζουμε μεθόδους και συναρτήσεις με ανεξάρτητους ελέγχους η καθεμιά. Τέλος, με τα ζητούμενα της άσκησης ήρθαμε σε πρώτη επαφή με το πως λειτουργεί γενικά μια διαχείριση μισθοδοσίας.

ΒΙΒΛΙΟΓΡΑΦΙΑ

docs.microsoft.com. (n.d.). Retrieved from https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/types/how-to-convert-a-string-to-a-number

dotnetbyexample.com. (n.d.). Retrieved from http://dotnetbyexample.com/calculate-age-from-dob-csharp/

hba.gr. (n.d.). Retrieved from https://www.hba.gr/Hebic/UplPDFs/D2015Gr/1.pdf