Day 05 .NET C# Exercises

1) *Lambda udtryk i kald af metode*
Åbn Console Application med følgende C# kode (udpak `lambda1.zip`)

```
    class Person
    {
        public string Name { get; set; }
        public int Age { get; set; }
    }

    class Program
    {
        static void Main(string[] args)
        {
            List<Person> persons = new List<Person>();
            persons.Add(new Person { Name = "Bent", Age = 25 });
            persons.Add(new Person { Name = "Susan", Age = 34 });
            persons.Add(new Person { Name = "Mikael", Age = 60 });
            persons.Add(new Person { Name = "Klaus", Age = 44 });
            persons.Add(new Person { Name = "Birgitte", Age = 17 });
            persons.Add(new Person { Name = "Liselotte", Age = 9 });

            // Indsæt kode her se spørgsmål nedenfor
            int i1=persons.FindIndex(...);
            ..

            Console.ReadLine();
        }
    }
```

Indsæt kode i parameteren til FindIndex i de fire tilfælde nedenfor. Undersøg selv erklæringen af FindIndex i documentationen for List<T>. *Du skal i hver af de fire tilfælde angive et lambda udtryk.*

a) Første Person i listen, persons, med alderen 44
b) Første Person i listen, persons, hvis navn starter med ’S’
c) Første Person i listen, persons, hvis navn indeholder mindst to ’i’
d) Første Person i listen, persons, hvis alder er lig med længden på navnet.

2) *ForEach i List*
Åbn Console Applicationen med følgende C# kode (udpak `lambda2.zip`):

```
using System;
using System.Collections.Generic;

class Program
{
    static void Main()
    {
        List<String> names = new List<String>();
        names.Add("Bruce");
        names.Add("Alfred");
        names.Add("Tim");
        names.Add("Richard");

        // Display the contents of the list using the Print method.
        names.ForEach(Print);
```

```
      Console.readLine();
    }

    private static void Print(string s)
    {
        Console.WriteLine(s);
    }
}
```

a) Undersøg typen på parameteren i ForEach metoden i List.
b) Ret eksemplet så det i stedet benytter lambda udtryk, dvs. erklæringen af den statiske Print metode skal slettes.

3) *Definition af metode der anvender Predicate delegate*

Åbn console application med følgende C# kode (udpak `lambda3.zip`)

```
    class Person
    {
        public string Name { get; set; }
        public int Age { get; set; }
        public int Score { get; set; }
        public bool Accepted { get; set; }

        public override string ToString()
        {
            string s = "Nej";
            if (Accepted) s = "Ja";
            return Name + ", Age=" + Age+", Score="+Score+", accepted="+s;
        }
    }

    public class Persons
    {
        List<Person> PersList = new List<Person>();

        public Persons()
        {
            Init();
        }

        private void Init()
        {
            PersList.Add(new Person { Name = "Bent", Age = 25, Score = 9 });
            PersList.Add(new Person { Name = "Susan", Age = 34, Score = 3 });
            PersList.Add(new Person { Name = "Mikael", Age = 60, Score = 8 });
            PersList.Add(new Person { Name = "Klaus", Age = 44, Score = 7 });
            PersList.Add(new Person { Name = "Birgitte", Age = 18, Score = 10 });
            PersList.Add(new Person { Name = "Liselotte", Age = 16, Score = 9 });
        }

        public void SetAccepted(...)
        {
        }

        public override string ToString()
        {
            string res = "[";
            foreach (Person p in PersList)
                res += p.ToString() + Environment.NewLine;
            res = res.Remove(res.Length - 2);
            res += "]";
```

```
            return res;
        }
    }

    public class Program
    {
        static void Main(string[] args)
        {
            Persons persons = new Persons();

            Console.WriteLine(persons.ToString());
            Console.WriteLine();

            persons.SetAccepted(p => p.Score >= 6 && p.Age <= 40);

            Console.WriteLine(persons.ToString());

            Console.ReadLine();
        }
    }
```

Færdiggør metoden SetAccepted så dens signatur og implementation passer med kaldet

```
persons.SetAccepted(p => p.Score >= 6 && p.Age <= 40);
```

som skal sætte `Accepted` til `true` for alle personer med `score>=6 og Age <=40` og `false` for alle andre.

Køre applicationen og se om det virker efter hansigten.

4) *Flere eksempler med FindAll på List<T>*
Åbn console application med følgende C# kode (udpak `lambda5.zip`)

```
public class Book
{
    public string Author { get; set; }
    public string Name { get; set; }
    public DateTime Published { get; set; }
}

class Program
{
    static void Main(string[] args)
    {
        //Create and fill a list of books
        List<Book> books = new List<Book> {
          new Book { Author="McConnell",Name="Code Complete",
                     Published=new DateTime(1993,05,14) },
          new Book { Author="Sussman",Name="SICP (2nd)",
                     Published=new DateTime(1996,06,01) },
          new Book { Author="Hunt",Name="Pragmatic Programmer",
                     Published=new DateTime(1999,10,30) } };

        var after1995 = books.FindAll(...);

        var between1991and1997 = books.FindAll(...);
    }
}
```

Færdiggør kaldet af de to FindAll og udskriv resultaterne.

5) *FindAll in Array class*

The Array Class declares the following FindAll method

```
public static T[] FindAll<T>(T[] array, Predicate<T> match)
```

Make a Console Application with the following array declaration

```
string[] names = {"Arne", "Mikael", "Erik", "Margrethe", "Stine",
"Jørn", "Morten", "Søren", "Hanne", "Gert", "Torben", "Karsten",
"Anne Dorte", "Kaj" };
```

Use Lambda expressions to call Array.FindAll to return new arrays:

a) All names starting with an 'A'
b) All names containing 'i'

Add

```
class Person
    {
        public string Name { get; set; }
        public int Age { get; set; }
    }
```

```
Person[] persons={ new Person { Name = "Bent", Age = 25 },
                   new Person { Name = "Susan", Age = 34 },
                   new Person { Name = "Mikael", Age = 60 },
                   new Person { Name = "Klaus", Age = 44 },
                   new Person { Name = "Birgitte", Age = 17 },
                   new Person { Name = "Liselotte", Age = 9 }
                 };
```

Use lambda expressions to call Array.FindAll to return new Arrays

c) Person[] young, which contains all Persons with and age below 30;
d) Person[] names5 which contains all Persons with a name of length 5.

6) *Simpel query*
Opret en Console Application med følgende erklæring i starten af Main metoden

```
int[] numbers = { 34, 8, 56, 31, 79, 150, 88, 7, 200, 47, 88, 20 };
```

For alle de følgende spørgsmål skal du skriver query med begge syntax former

a) Opret en query der returnerer alle tal i numbers der har 2 cifre og sorteret i stigende orden

b) Det samme men sorteret i faldende orden

c) Det samme som a), men resultatlisten skal indeholde string objekter: ”20”, ”31”, ”34” …

d) Det samme som a), men resultatlisten skal indeholde

```
20 lige
31 ulige
```

...

Hjælp: husk du kan skrive et udtryk som (x<b?c1:c2)

e) Opret en query der returnerer en liste med anonyme typer med første felt som tallet og næste felt som true eller false til at angive om tallet er lige eller ej, Felterne skal hedde "Tal" og "Lige".

f) Hvis du ikke har skrevet en metode med en foreach der kan udskrive resultaterne af de enkelte queries, så gør det og brug den i din løsning.

7) *Simpel query*
Opret en Console Application og tilføj følgende erklæring og metode

```
public class Customer
{
    public string CustomerID { get; set; }
    public string City { get; set; }

    public override string ToString()
    {
        return CustomerID + "\t" + City;
    }
}

static IEnumerable<Customer> CreateCustomers()
{
    return new List<Customer>
    {
        new Customer { CustomerID = "ALFKI", City = "Berlin"    },
        new Customer { CustomerID = "BONAP", City = "Marseille" },
        new Customer { CustomerID = "CONSH", City = "London"    },
        new Customer { CustomerID = "EASTC", City = "London"    },
        new Customer { CustomerID = "FRANS", City = "Torino"    },
        new Customer { CustomerID = "LONEP", City = "Portland"  },
        new Customer { CustomerID = "NORTS", City = "London"    },
        new Customer { CustomerID = "THEBI", City = "Portland"  }
    };
```

For alle de følgende spørgsmål skal du skriver alle queries med begge syntax former

a) Skriv en query der finder alle Customer fra London
b) Skriv en query der finder CustomerId på alle Customer der ikke er fra London

8) *Group by*
Lav en Console Application med følgende erklæring i Main

```
string[] words = { "blueberry", "chimpanzee", "abacus", "banana",
                   "apple", "cheese" };
```

Skriv en query med Group by i begge syntakser, der lister alle navn grupperet efter første bogstav.

9) *Group by*
Lav en Console Application med følgende erklæring i Main

```
int[] numbers = { 5, 14, 1, 33, 9, 8, 20, 26, 44);
```

Skriv en query med Group by i begge syntakser, der grupperer tallene efter deres rest ved division med 5.