Programmering 1 - Inför prov

Övningar och betygsnivåer inför prov i algoritmer.

Betyg E

Kunna hantera fält och samlingsklasser (List<int>, List<double>, List<string> etc).
Förstå skillnaden mellan ett fält och en samlingsklass t.ex. List<>.
Hantera enkla objekt från .NET, t.ex. Random-klassen.
Hantera sök- och sorteringsalgoritmer med viss hjälp.

Betyg C

Hantera sök- och sorteringsalgoritmer utan större hjälp.
Känna till .NET-ramverket bättre, t.ex. metoder på samlingar (listor/fält).
Känna till hur stacken fungerar vid metodanrop.

Betyg A

Hantera olika sök- och sorteringsalgoritmer utan hjälp.
Känna till rekursivitet och gärna kunna någon rekursiv variant av algoritm.
Förstår begreppet komplexitet gällande algoritmer (“the big O”).
Kunna analysera algoritmer och bedöma effektivitet/komplexitet för egna algoritmer men även utifrån exempel.
Känna till hur stacken fungerar vid metodanrop och begreppet Stack Overflow.

Övningar

Övning 1 (E-nivå)

1. Skapa en lista (eller fält) med 100 platser och fyll med slumpvis data mellan 0 och 100.
2. Hitta det största talet i listan/fältet.
3. Hitta det minsta talet i listan/fältet.
4. Beräkna summan av alla tal i listan/fältet.
5. Hitta ett specifikt värde i listan/fältet. Om värdet inte finns i listan/fältet så ska det upptäckas på något vis.
6. a. Sortera listan/fältet genom att använda en valfri sorteringsalgoritm som implementeras **med** förlaga (t.ex. anteckningar om hur den fungerar + pseudokod).

Övning 1 (C-nivå)

6. b. Sortera listan genom att använda en valfri sorteringsalgoritm som implementeras **utan** förlaga.
7. Skriv sorteringsalgoritmen i en separat metod i ditt program som du sedan använder.
8. Beräkna medelvärdet av talen i listan/fältet.

9. Beräkna medianen av talen i listan/fältet.
10. Känna till och kunna använda de inbyggda metoderna Min(), Max() och Average()

Övning 1 (A-nivå)

11. Sortera listan genom att använda en algoritm som är effektivare än Bubble Sort (t.ex. Insertion sort).
12. Med sorterad lista/fält kunna hitta ett specifikt värde på ett effektivt sätt, t.ex. med Binär sökning.
13. Hitta alla primtal som finns i listan/fältet och skriva ut dem.

Övning 2 (E-nivå)

Övningar i att behärska och skriva egna metoder.

1. Skriv en metod som skriver ut ditt namn. Metoden ska ha varken in- eller returparameter. Använd och testa din metod!
2. Skriv en metod som returnerar ett slumptal mellan 1 och 6. Kalla den **RollDice()**. Använd metoden i ditt program och visa att den fungerar.
3. Skriv en metod som returnerar det största av två värden som skickas in. Kalla den **LargestOf(int a, int b)**. Visa att metoden fungerar genom att skriva kod som testar olika fall.

Övning 2 (C-nivå)

4. Skriv en metod som skriver ut text i Console med en viss färg. Inparametrar bör vara färg samt den text som ska skrivas ut. Skriv sedan ut något färgglatt!
5. Skriv en metod som räknar antalet ord i en string (mellanslag skiljer mellan ord). Inparameter är strängen och utparameter är en int (antalet ord). Visa att metoden fungerar.
6. Skriv en metod som hittar det längsta ordet i en sträng. Metoden ska returnera det längsta ordet (string) och har texten som inparameter. Visa att metoden gör sitt jobb.

Övning 3 (A-nivå)

Övning i grundläggande rekursivitet.

1. Kunna skriva om följande loop till en rekursiv loop:

```
for(int i=10; i>=0; i--)
{
    Console.WriteLine(i);
}
```

2. Skriv om följande rekursiva program till en iterativ loop:

```
static void Main(string[] args)
{
    SkrivUt(0);
```

```
        }

        static void SkrivUt(int i)
        {
            if (i >= 10)
                return;
            Console.WriteLine(i);
            SkrivUt(i + 1);
        }
```

Övning 4 (A-nivå)

Analysera följande kod och bestäm Ordo (O) för alla. Gärna med motivering. Ange bästa-, sämsta- och medelfallet om det finns skillnader mellan dessa.

#1

```
var M = A[ 0 ];

for ( var i = 0; i < n; ++i ) {
      if ( A[ i ] >= M ) {
            M = A[ i ];
      }
}
```

#2

```
bool exists = false;
for (int i = 0; i < n; i++ ) {
      if ( A[i] == value ) {
            exists = true;
            break;
      }
}
```

#3

En algoritm som letar om det finns någon dublett av tal i listan/fältet.

```
bool duplicate = false;
for ( int i = 0; i < n; ++i ) {
      for ( int j = 0; j < n; ++j ) {
            if ( i != j && A[i] == A[j] ) {
                  duplicate = true;
                  break;
            }
      }
      if ( duplicate ) {
```

```
                break;
        }
    }
```