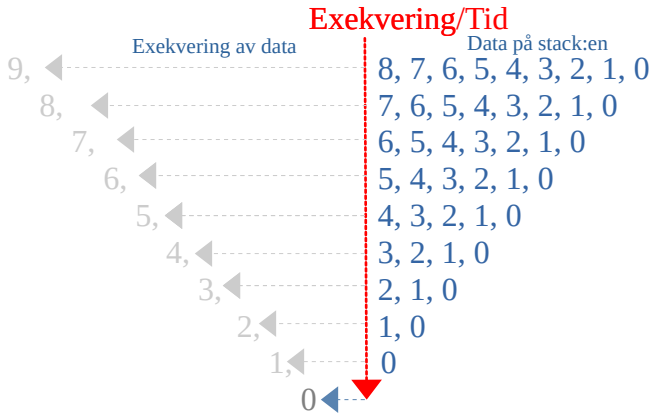
Fråga 1. *”Hur fungerar stacken och heapen? Förklara gärna med exempel eller skiss på dess grundläggande funktion ”*

1.1 Stack
Stackens struktur kan liknas vid en stapel med en bestämd ordning för exekvering, data kan enbart nås när tidigare data har exekverats. Det går även att likna stacken vid en behållare med slimmad passform t.ex. ett rör med tennisbollar. Bollen som stoppats ner först går inte plocka ut förrän bollarna som ligger över har plockats ut. I det här exemplet symboliseras stacken med en behållare där tennisbollarna representeras av siffror, se figur 1 nedan.

9, 8, 7, 6, 5, 4, 3, 2, 1, 0

Figur 1. Stack/behållare med data/tennisbollar.

För att nå data i en stack/tennisboll i behållare som placerats på plats ”0” måste det tidigare platserna 9-1 arbetats igenom(hämtats). I figur 2 nedan visas hur behållaren/stacken töms/arbetas igenom under exekverings-/bollhämtnings- tiden.



Figur 2. Exekvering av data från stacken

I stacken lagras värdetyper, adresser samt funktionsanropningar. En metod som består av värdetyper som deklarerats i metoden resulterar i lagring på stacken.

1.2 Heap
Heap:ens struktur kan liknas vid en hög med papper. I Heap:en kan all data nås för användning utan hänsyn till var i stacken önskad data för användning är placerad. Se skiss nedan i figur 3.

```
      0
    1   2
  3   4   5
6   7   8   9
```

Figur 3. Skiss av heap-struktur.

I heap:en lagras referenstyper men även värdetyper. Data som allokeras i heapen rensas ej automatiskt efter exekvering, i .NET-ramverket sker detta dock med hjälp av GC(garbage collector). Om vi använder oss av exemplet för stacken med en metod som består av värdetyper och istället för att deklarera en av värdetyperna i en klass så kommer typen att placeras i heap:en.

Fråga 2. *”Vad är Value Types respektive Reference Types och vad skiljer dem åt?”*

2.1 Värdetyp
Värdetyper är exempelvis data i form av en integer som upptar minnesutrymme.

2.2 Referenstyp
En referenstyp refererar till en adress minnet genom en pekare där data är placerad.

Fråga 3. *”Följande metoder (se bild nedan) genererar olika svar. Den första returnerar 3, den andra returnerar 4, varför?”*

```
public int ReturnValue()
{
    int x = new int();
    x = 3;
    int y = new int();
    y = x;
    y = 4;
    return x;
}

0 references
public int ReturnValue2()
{
    MyInt x = new MyInt();
    x.MyValue = 3;
    MyInt y = new MyInt();
    y = x;
    y.MyValue = 4;
    return x.MyValue;
}
```

3.1 Metod ReturnValue()
I metoden sätts x till tre och behåller sitt värde som slutligen returneras.

3.2 Metod ReturnValue2()
x.MyValue och y.MyValue delar samma referens som först sätts till 3 och sedan till fyra.