Fredrik Sjöberg – fresj034@student.liu.se
Laboration 2 – frågor

1. **Förklara vad Cocoa Touch är för någonting.**

Cocoa touch är ett UI ramverk för utveckling av applikationer till iOS. Fokus för Cocoa Touch ligger på touch-baserade interface.

2. **Objective-C använder sig av "message passing" för att skicka meddelanden till objekt (ungefär detsamma som ett metodanrop). Vad innebär detta i praktiken? Vet man med säkerhet att en metod finns vid kompileringstillfället? Hur skiljer sig message passing från ett statiskt metodanrop, som till exempel används i C++, C#, Java, python, med flera? Inga detaljer erfordras, bara den grundläggande skillnaden.**

I praktiken innebär detta att om man skickar ett meddelande till ett objekt som det inte kan svara på kan det skickas vidare i en kedja tills det kommer fram något som kan svara på det. Det blir alltså inga kompileringsfel utan det kan köras ändå, det kommer endast returneras ett undantag om meddelandet inte får något svar. Objective-C skiljer sig från andra språk som t.ex. C++ då den är högst dynamisk och tillåter tilläggning av metoder under runtime medans definitionen av en class i C++ inte kan ändras under runtime.

3. **Vad är ett prokotoll för någonting i Objective-C? Hur används ett sådant?**

Ett protokoll i Objective-C är ett interface (som det är känt som i andra programmeringsspråk). Protokoll kan inkludera både instansmetoder, klassmetoder och egenskaper. Ett protokoll används för metoder och egenskaper som är oberoende av en specifik klass.

4. **Vad är ett så kallat "block" i Objective-C-sammanhang? Hur används ett block? Finns det några begränsningar för när man kan använda block?**

Ett block i Objective-C används för att skapa distinkta segment av kod som kan skickas runt till metoder/funktioner som om de vore värden. Block är Objective-C Objekt. Ett block definieras med hjälp av ^ symbolen:

```
^{
    NSLog(@"This is a block");
}
```

5. **Förklara vad en kategori ("category" eller ibland även kallat "addition") är i Objective-C och vad dessa är bra till. Har du stött på något liknande i något annat programmeringsspråk?**

Categories tillåter en att lägga till metoder/funktionalitet till klasser som redan existerar. Det kan vara bra att kunna lägga till funktionalitet till klasser istället för att behöva skapa objekt varje gång en viss funktion behövs.

6. **Vad är en "property" för något? Hur lägger man till en sådan till en klass (vilka saker behöver göras)? Det finns ett antal attribut man kan ge en property (nonatomic, strong, ...) - förklara kort vad dessa är till för.**

Properties är publika instansvariablar, som innebär att externa klasser får tillgång till den specifika propertyn. Properties kan ta olika attribut som till exempel nonatomic vilket innebär att flera

threads kan använda propertyn. Om nonatomic inte anges anses propertyn ha attributet atomic vilket låser flera threads att använda propertyn samtidigt. Strong är ett annat attribut som skapar en ägarrelation av ett objekt. Det finns även ett flertal andra attribut som readonly, copy osv.

7. **Förklara vad skillnaden är mellan en stark och en svag referens till ett objekt. När ska man använda en stark respektive svag referens?**

En stark referens till ett objekt innebär att du äger objektet du refererar till med propertyn/variabeln, detta innebär att propertyn inte kommer förstöras förens utvecklaren anger den till nil.

En svag referens innebär att utvecklaren inte vill ha kontroll över objektets livstid. Objektet lever endast vidare om åtminstone ett annat objekt håller en stark referens till den. När den inte har en stark referens förstörs objektet och propertyn blir automatiskt nil.

8. **Med starka referenser kan man få något som kallas en retain cycle. Vad är det? Vilka problem leder det till? Hur ska man göra för att undvika retain cycles? Förklara svaren och visa ett fall av en retain cycle med hjälp av ett kodexempel.**

En retain cycle är en situation där objekt A bevarar (retains) objekt B, och objekt B bevarar objekt A samtidigt. Nedan är ett exempel på en retain cycle

```
@class Child;
@interface Parent : NSObject {
    Child *child; // Instance variables are implicitly __strong
}
@end
@interface Child : NSObject {
    Parent *parent;
}
@end
```

En retain cycle innebär att det är omöjligt för objekten att releasea och då hålls dom kvar. En lösning för detta är att använda svaga properties istället.

9. **Vad är det för skillnad på NSArray, NSDictionary och NSSet? Vad är respektive datastruktur lämplig att användas till? Finns det några andra liknande datastrukturer?**

NSArray = håller objekt i sorterad ordning och är lämplig för att ha en sorterad lista
NSSet = osorterad array som är bra när listan inte behöver vara sorterad utan bara spara flera objekt i en lista
NSDictionary = håller objekt som nyckel till ett värde och bör användas om man vill koppla en nyckel till ett värde.

10. **De flesta datastrukturer finns i två varianter: en muterbar (mutable) och en icke-muterbar (immutable). Ett exempel på en muterbar struktur är NSMutableArray medan den icke-muterbara heter NSArray. Förklara vad skillnaden är mellan dessa begrepp (var generell i ditt svar).**

NSArray är inte muterbar och kan därmed inte modifieras medans en NSMutableArray är en muterbar array som kan lägga till objekt i efterhand.

11. **När du utvecklat vanliga desktop-applikationer har du säkert använt dig av spårutskrifter, för att till exempel se att du kommer till en viss punkt i koden eller undersöka en variabels värde. Liknande möjligheter finns även när du utvecklar på iPhone eller iPad - vad heter funktionen du använder? Hur gör du för att formatera utskrifter?**

    Funktionen heter NSLog och fungerar som de flesta spårutskrifter, genom att skriva ut text till konsolen. Olika datatyper formuleras i koden, alltså om en integer ska skrivas ut:

```
int num=100;
NSLog(@"%i", num);
```