Laboration 2.2: ADT-erna kö, stack och lista.

I denna uppgift ska ni göra implementationer för ADT:erna **Stack**, **Queue** och **Set**. Var och en av ADT:erna går att implementera som en **array** eller som en **länkad lista**. Ni får välja implementation för alla tre, men ni måste implementera **minst en ADT som en länkad lista** och **minst en som en array**. Det är dock upp till er vilken ni väljer för respektive ADT.

Börja att ladda ner `QueueStackSet.zip` från Canvas. Zippa upp dessa filer i samma mapp som c/h-filerna för länkade listan. Lägg sedan till dessa filer till projektet (project/add.. I visual studio) där ni har er länkade lista (alltså projektet ni skapade för labb 2.1). Ni ska alltså inte göra ett nytt projekt. Det är viktigt att filerna ligger i samma mapp som list-filerna.

I `test.c` finns testfunktioner för var och en av ADT:erna. Dessa får inte ändras och ska fungera oavsett implementation. För att köra testerna, lägg till `#include "test.h"` till filen `test_list.c`. Lägg sedan till ett anrop till funktionen `testADT()` från `main()`.

Syftet med denna uppgift är att ni ska förstå hur en abstrakt datatyp (ADT) kan fungera likadant oavsett underliggande implementation (konkret datatyp).

Uppgift 2.1: Stack

- Öppna filen `stack.h`
- Välj om ni vill implementera stacken som en array eller länkad lista.
- Gå till "1. Välj implementation" i `stack.h` och avkommentera en av `#define`-raderna
- Titta på avsnittet 2A eller 2B i koden beroende på implementation. Typen 'Stack' kommer alltså antingen att vara er länkade lista eller en struct som innehåller en array. Ändra inget i dessa avsnitt.
    - Väljer ni en arrayimplementation kan det bli aktuellt att lägga till fler medlemmar än arrayen i structen. Se avsnitt 2B i koden.
- Gå ner till "3. Interface" och studera interfacet till stacken. Ingen av dessa funktionsdeklarationer får ändras på något vis. Se till att ni förstår vad varje funktion ska göra.
- Öppna filen `stack.c`. Implementera varje funktion med hjälp av en array eller lista. Verifiera pre- och postconditions som står som kommentarer med asserts.
- Öppna filen `test.c` gå till `testADT()` och kommentera bort allt utom `testStack()`. Lägg sedan till ett anrop till `testADT()` i main(), som ligger i `test_list.c` (och ev. Kommentera bort anropet till menyn). Om ingen `assert()` aktiveras när ni kör `testStack()` så fungerar er stack som den ska! Om en `assert()` aktiveras så har ni gjort ett fel någonstans. Se `testStack()` för att se var felet uppstod och varför. Ni får naturligtvis inte göra ändringar i `testStack()` eller någon annan testfunktion.

Viktigt!
Om ni använder länkad lista som implementation så får ni enbart använda funktionerna i `list.h` för att använda listan. Ni ska ej behöva känna till hur listan är implementerad. Ni ska alltså **inte** röra några struct-medlemmar (t.ex `data` eller `next`).

Tänk på att varje funktion oavsett om ni använder en länkad lista eller en array kommer att bli väldigt kort och innehålla enbart ett fåtal rader.

Uppgift 2.2: Kö

Upprepa samma procedur som i Uppgift 2.1, men använd istället filerna `queue.h`, `queue.c` och funktionen `testQueue()`.

Uppgift 2.3: Set

Upprepa samma procedur som i Uppgift 2.1, men använd istället filerna `set.h`, `set.c` och funktionen `testSet()`.