Opgave 1 (Typer, Aritmetik, Logik, I/O)

1. Hent en kopi af **Sandbox**-projektet fra **GitHub**.
2. Åbn projektet i **Visual Studio**, og kør programmet (tryk F5 eller klik på den lille grønne trekant). Programmet skulle gerne udskrive ***"Hello, World!"*** på skærmen.
3. Åbn filen **InsertCodeHere.cs** (dobbelt-klik på den i **Solution Explorer** vinduet). Slet den ene linje mellem de to grønne kommentar-linjer.
4. Gør følgende mellem de to kommentar-linjer:
   - Definér en variabel af typen **int**, med navnet **cash**. Sæt den lig med 12.
   - Definér en variabel af typen **double**, med navnet **price**. Sæt den lig med 11.95.
   - Definér en variabel af typen **bool**, med navnet **canBuy**. Sæt den lig med et logisk udtryk, som kun er sandt hvis **cash** er større end **price**.
   - Udskriv en **string**, som lyder ***"I can afford it: "***, efterfulgt af værdien af **canBuy**.
   - Kør programmet. Med de værdier der er angivet ovenfor, burde der blive udskrevet: ***"I can afford it: True"***
   - Ret værdien af **price** til 12.95, og kør programmet igen. Nu burde programmet udskrive: ***"I can afford it: False"***

Opgave 2 (Control statements)

1. Hent en kopi af **Sandbox**-projektet fra **GitHub**. Hvis du allerede har hentet **Sandbox**-projektet, kan du springe dette skridt over.
2. Åbn projektet i **Visual Studio**, og åbn filen **InsertCodeHere.cs**. Slet alt mellem de to grønne kommentar-linjer.
3. Gør følgende mellem de to kommentar-linjer:
   - Definér en variabel af typen **int**, med navnet **tvScreenSize**. Sæt den lig med 55.
   - Definér en variabel af typen **bool**, med navnet **tvOLEDTech**. Sæt den lig med **false**.
   - Udskriv en **string**, som følger denne specifikation
     - Hvis **tvScreenSize** er mindst 50 og **tvOLEDTech** er **true**, udskriv ***"That's a very nice TV!"***, ellers
     - Hvis **tvOLEDTech** er **true**, udskriv ***"That's a fancy TV!"***, ellers
     - Udskriv ***"That's a…TV…"***
   - Kør programmet. Med de værdier der er angivet ovenfor, burde der blive udskrevet: ***"That's a…TV…"***
   - Prøv at ændre værdierne af **tvScreenSize** og **tvOLEDTech**, så de andre tilfælde også bliver afprøvet.
4. Slet koden du har skrevet for at løse 3. Gør nu følgende mellem de to kommentar-linjer:
   - Skriv en **for**-loop, som for alle tal mellem 0 og 30 udskriver følgende (bemærk, at på den plads hvor der står ***(tal)*** skal selve tallet stå):
     - Hvis tallet kan deles med fire, udskrives ***"(tal) går op i 4"*** (f.eks. ***"8 går op i 4"***)
     - Hvis tallet ikke kan deles med fire, udskrives ***"(tal) går ikke op i 4"*** (f.eks. ***"11 går ikke op i 4"***)
   - Kør programmet, og check om udskriften er korrekt.

Opgave 3 (Datastrukturer)

1. Hent en kopi af **Sandbox**-projektet fra **GitHub.** Hvis du allerede har hentet **Sandbox**-projektet, kan du springe dette skridt over.
2. Åbn projektet i **Visual Studio**, og åbn filen **InsertCodeHere.cs**. Slet alt mellem de to grønne kommentar-linjer.
3. Gør følgende mellem de to kommentar-linjer:
   - Definér en variabel af typen **List<int>** kaldet **minListe**, og sæt variablen til at referere til en ny liste, således: **List<int> minListe = new List<int>();**
   - Indsæt tallene 8, 14, 3, -2 og 47 i **minListe**.
   - Brug en **foreach**-loop til at skrive alle tallene i **minListe** ud.
   - Brug en **for**-loop til at skrive tallene i **minListe** ud i omvendt rækkefølge.
4. Slet koden du har skrevet for at løse 3. Gør nu følgende mellem de to kommentar-linjer:
   - Definér en variabel af typen **Dictionary<string, int>** kaldet **mineTests**, og sæt variablen til at referere til en ny Dictionary, således: **Dictionary<string, int> mineTests = new Dictionary<string, int>();**
   - Indsæt disse (nøgle,værdi)-par i **mineTests**: (”Matematik”, 65), (”Idræt”, 80), (”Biologi”, 90) og (”IT”, 75).
   - Brug en **foreach**-loop til at skrive alle nøglerne i **mineTests** ud.
   - Beregn gennemsnittet af værdierne i **mineTests**, og skriv resultatet ud.
   - Prøv at slette et par elementer fra **mineTests** ved at kalde metoden **Remove**, og regn efterfølgende det nye gennemsnit af værdierne ud.

Opgave 4 (Klasse-definition, simpel)

1. Hent projektet **RepetitionExercises** fra **GitHub**.
2. Åbn projektet i **Visual Studio**, og åbn filen **Contact.cs**. Denne fil rummer starten på en klasse-definition for klassen **Contact**.
3. Tilføj et antal instance fields til **Contact**. Disse instance fields skal kunne rumme information om kontaktens:
    a. Navn
    b. Fødselsår
    c. E-mail adresse
    d. Om kontakten er et familiemedlem eller ej.
4. Tilføj en constructor til **Contact**, således at alle instance fields får en initiel værdi, når der skabes et **Contact**-objekt. Det skal gælde at:
    a. Navn, fødselsår og familiemedlem-eller-ej skal være parametre til constructoren
    b. E-mail sættes altid til denne initielle værdi: ukendt@ukendt.dk
5. Tilføj et antal properties til **Contact**, svarende til de definerede instance fields. Det skal gælde at:
    a. Navn, fødselsår og familiemedlem-eller-ej kan kun læses, ikke ændres
    b. E-mail kan både læses og ændres
6. Tilføj en metode kaldet **PrintSummary** til **Contact**. Den skal udskrive alle oplysninger om denne kontakt på skærmen.
7. Åbn **Program.cs**, og skriv noget kode som skaber og benytter et par **Contact**-objekter. Koden skal skrives i metoden **Main**, lige over kaldet af **KeepConsoleWindowOpen**.

Opgave 5 (Klasse-definition med association)

1. Hent projektet **RepetitionExercises** fra **GitHub**.
2. Åbn projektet i **Visual Studio**. Hvis du allerede har hentet projektet, kan du springe dette skridt over.
3. Åbn filerne **SmartTV.cs**, **Speaker.cs**, **BluRayPlayer.cs** og **HomeTheater.cs** (de ligger i mappen **HomeTheaterClasses**). De tre klasser **SmartTV**, **Speaker** og **BluRayPlayer** skal modellere komponenter i et ”home theater” (HT) system. Dan dig et overblik over de tre klasser.
4. Klassen **HomeTheater** er (næsten) tom. Gør følgende:
    a. Definér nogle instance fields, således at klassen kommer til at modellere et HT-system med: et SmartTV, en BluRayPlayer og to Speakers (venstre og højre).
    b. Initialisér de nye instance fields i constructoren
    c. Implementér alle de metoder, som er tomme i klassen. De skal virke som angivet i kommentaren til hver metode.
5. Når klassen er færdig, kan den testes ved at kalde **HomeTheaterTest.Run()** fra **Main** i **Program.cs**. Det forventede output er angivet i **HomeTheaterTest.cs**.

Opgave 6 (Klasse-definition med nedarvning)

0. *Vi arbejder videre med projektet **RepetitionExercises**.*
1. Der er en del fælles elementer i de tre klasser **SmartTV**, **Speaker** og **BluRayPlayer**. Benyt klassen **HomeTheaterDevice** som base-klasse for de tre klasser. Mere specifikt skal du:
    a. Lade hver af de tre klasser arve fra **HomeTheaterDevice**
    b. Fjerne/omskrive kode i de tre klasser, således at alt fælles kode ligger i base-klassen. Vær opmærksom på de virtuelle metoder **HandleOn** og **HandleOff**.
2. Når klasserne er omskrevet, kan du igen benytte metoden **HomeTheaterTest.Run()** til at teste, om applikationen stadig virker som den skal

Opgave 7 (Klasse-definition med interface)

0. *Vi arbejder videre med projektet **RepetitionExercises**.*
1. Der er stadig den begrænsning i **HomeTheater**, at den definerer et fast setup for et HT-system. Overvej, hvordan vi kan løsne denne begrænsning.
2. Kig på interfacet **IHomeTheaterDevice** – hvordan kan det benyttes til at lave en løsere kobling mellem **HomeTheater** og specifikke devices?
3. Hvad sker der med ansvaret for at definere afhængigheder mellem **HomeTheater** og specifikke devices, i forhold til den løsning du er kommet frem til i 2.?
4. Hvis vi f.eks. indfører et instance-field af typen **List<IHomeTheaterDevice>**, bliver det ikke helt så oplagt, hvordan man skal implementere en metode som f.eks. **IncreaseVolume**. Overvej, hvordan man kan holde rede på, hvilke devices det kunne give mening at kalde **IncreaseVolume** på (Hint: Interfaces…).