# Vejledning til løsningsforslaget

## Delopgave 1

Laver et C# console projekt. Tilføjer en klasse Kernel32Wrap hvor jeg definerer den eksterne funktion Beep. Bruger attributten DllImport til at angive at Beep findes i en dll med navnet Kernel32, og setter SetLastError.

I Main kalder jeg så Beep. Bemærk koden hvor jeg forsøger at fremprovokere en fejl. Det lykkes ikke på Windows 7 og 8, men hjælpeklassen til at udskrive en relevant fejltekst, kan være meget nyttig i andre sammenhænge.

Til sidst ”spiller” jeg en skala. Hjælpeklassen BeepMusic kan bruges til at spille simple melodier (hvis du kender noderne), men hvis du for alvor vil arbejde med music på PC’en skal du nok bruge MIDI interfacet.

## Delopgave 2

Der laves et console-projekt ”Lab2”, og til dette tilføjes dll’en fra opgave 01.1 (dette er ikke nødvendigt, men gør at jeg via dens properties kan specificere at den skal kopieres til output-dir automatisk).

Denne gang laver jeg ingen wrapper-klasse, men har mine ekstern erklæringer direkte i samme klasse. Da det er ”plain” c-funktioner skal CallingConvention sættes til Cdecl.

Bemærk at funktionen myAddString ikke kan anvendes (uden store krumspring), da det ikke er nogen indbygget konvertering mellem .Nets string type og C++’s string type.

Det store problem i opgaven er at modtage et array of char fra C++. IntPtr og Marshal-klassen kan bruges:

IntPtr ptr = myAddStr(str1, str2);   
string strRes = Marshal.PtrToStringAnsi(ptr);

Dette efterlader os med et problem: prt vil stadig pege på noget unmanaged memory (som i dll’en er allokeret med C++’s new operator). Som koden er skruet sammen i dette løsningsforslag, er der ingen umiddelbar løsning. Det skal laves en ændring i dll’en for at vi kan frigive den allokerede memory fra C#. Der er 2 muligheder:

**Alternativ 1:**

Vi kan udvide dll med en DeletePtr funktion, som vi så kan kalde fra C#, når vi ønsker at frigive det unmanaged char array (efter at vi har kopieret strengen). Dette løsningsforslag kan ses i Solution-mappen Lab2\_Alt1.

**Alternativ 2:**

I stedet for at bruge c++’s new operator kan man i dll’en allokere memory med windows API-funktionen LocalAlloc. Det gode ved den er, at denne hukommelse kan frigives med Marshal. FreeHGlobal som umiddelbart kan kaldes fra C#. Dette løsningsforslag kan ses i Solution-mappen Lab2\_Alt2.

## Delopgave 3

Laver et nyt C++ - dll projekt med navnet Lab3\_dll. Kopierer koden fra filen DllWithCallBack.h over i filen Lab3\_dll.h, og laver de fornødne tilretninger. Kopierer koden fra DllWithCallBack.cpp over i filen Lab3\_dll.cpp, laver de fornødne tilretninger og bygger projektet.

Laver et nyt C# - console projekt med navnet Lab3.

Definerer signaturer til dll-funktionen SomeUnmanagedFunctionUsingCallback:

[DllImport("Lab3\_Dll.dll", CallingConvention = CallingConvention.Cdecl)]

public static extern void SomeUnmanagedFunctionUsingCallback(double x, CallBackDelegate callback);

// Definition of delegate used for callback function

public delegate double CallBackDelegate(double x);

Hvorefter jeg kalder funktionen:

CallBackDelegate **theFunctionDelegate** = new CallBackDelegate(Math.Sin);

SomeUnmanagedFunctionUsingCallback(Math.PI / 2.0, **theFunctionDelegate**);