Part 1: Disassembling code, going backwards, converting an executable back to Assembly Language.

To do: I have taken the executable version of the three programs you have written recently and renamed them A, B and C. These programs were “Hello world”, “Typing tutor” and “Floating point calculation”.

```
D:\Aenvironment\nasm>ndisasm -e217h -b16 C:\program1\MASM\A.exe
00000000  BB8E00        mov bx,0x8e
00000003  8B17          mov dx,[bx]
00000005  7407          jz 0x11
00000007  8AF4          cmp dl,0x24
0000000A  B402          mov ah,0x2
0000000C  CD21          int 0x21
0000000E  EBF3          jmp short 0x3
00000010  43            inc bx
00000011  90            nop
00000012  B44C          mov ah,0x4C
00000014  CD21          int 0x21
00000016  0D8E65        add [bx+si+0x65],cl
00000019  6C            insb
0000001A  6C            insb
0000001B  66            outsw
0000001C  2F            and [bx+0x6F],dh
0000001F  726C          jc 0x8d
00000021  64            fs
00000022  24            db 0x24
```

```
D:\Aenvironment\nasm>ndisasm -e217h -b16 C:\program1\MASM8.B.exe
00000000  B408          mov ah,0x8
00000002  CD21          int 0x21
00000004  B0A8          mov al,0xA8
00000006  8AFD          mov dh,dl
00000008  B007          mov al,0x7
0000000A  7413          jz 0x1e
0000000C  8AE3          sub dl,bl
0000000E  70F0          jc 0x0
00000010  8AF9          cmp dx,0x9
00000013  77EB          ja 0x0
00000015  8BC230        mov dx,0x30
00000018  B402          add dh,0x30
0000001A  CD21          int 0x21
0000001C  EBE2          mov ah,0x2
0000001E  B44C          jmp short 0x0
00000020  CD21          mov ah,0x4C
00000022  CD21          int 0x21
```

```
D:\Aenvironment\nasm>ndisasm -e217h -b16 C:\program1\FloatingPointCalc.asm
00000000  B80000        mov ax, 0
00000003  DB2D          fldpi
00000005  D94508        fld dword [bp+8]
00000008  DEC1          faddp st1
0000000A  DD1E2000      fstp dword [0x2000]
0000000E  B402          mov ah, 0x2
00000010  CD21          int 0x21
00000012  B44C          mov ah, 0x4C
00000014  CD21          int 0x21
```

Part 2: Ask the compiler to generate Assembly Language as well as an Executable

To do: Create a New Visual Studio C++ project called "To_ASM" and save it on either your X drive or C:\temp.

```
#include "stdafx.h"
#include <stdio.h>
int _tmain(int argc, _TCHAR* argv[])
{
register int total=0;
for(register int i=0;i<=20;i++)
{
total+=i;
}
printf("%d",total);
getchar();
return 0;
}
```

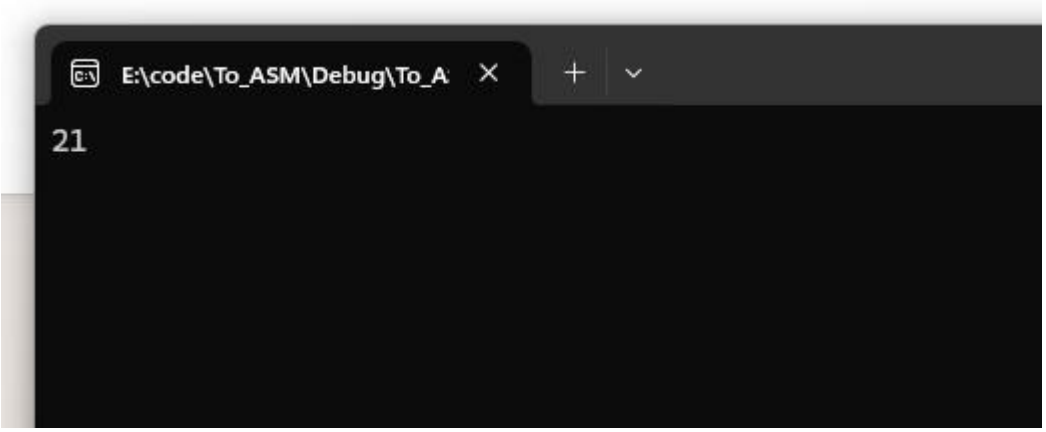
```
E:\code\To_ASM\Debug\To_A
210
```

```
00003d   8B45F8        mov eax, DWORD PTR [ebp-8]
000040   0345EC        add eax, DWORD PTR [ebp-20]
000043   8945F8        mov DWORD PTR [ebp-8], eax
```

Part 3: Investigation of Pointer

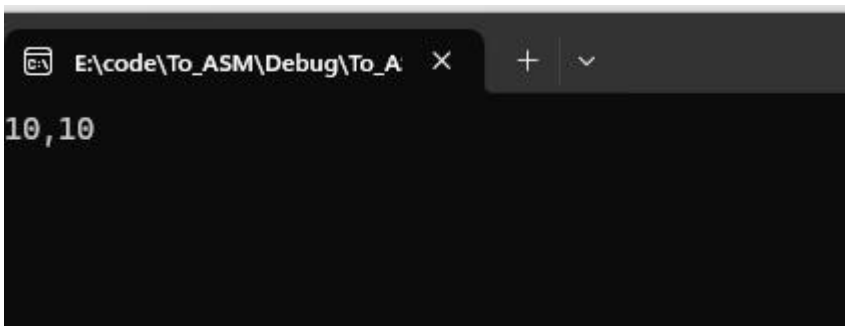
To do: Consider the following program and talk through the various operators with a friend so as to make a prediction of the final run time value of the variable y.

```
#include "stdafx.h"
#include <stdio.h>
int _tmain(int argc, _TCHAR* argv[])
{
int *address; // Create a pointer that can store an address (location in memory)
int x=10; // Create an integer variable called x containing value 10
int y=0; // y is an int with an initial value 0
int z[3]={5,7,11}; // Create an array of three integers
address=&x; // Let address store the value of the address of x in memory
y=*address; // y is assigned the value of whatever is pointed to by address
address =&z[0]; // address is set to the location in memory of the first value
y+=*(address+2); //y is incremented by the value pointed to at two items above address
printf("%d\n",y); // prints y to screen (y=????)
// Wait for enter to be pressed before terminating
while(getchar()!=10); // Clear buffer of previous <ret>
while(getchar()!=10); // Wait for a new <ret>
return 0;
}
```




```
#include "stdafx.h"
#include <iostream>
using namespace std;
// Create a class (object) called Sphere, this is a
// description of the object, not an instance of the object.
// It is the cookie cutter not the cookie.
class Sphere
{
public: float radius;
public: float surface_area();
// Class constructor runs when a new instance of "sphere" is created.
```

```
Sphere()
{
radius=1.0f;
}
// In line function belonging sphere
float diameter()
{
return(2.0f*radius);
}
};
// External method used by sphere
float Sphere::surface_area()
{
float area=4.0f*3.14159*(radius*radius);
return(area);
}
int _tmain(int argc, _TCHAR* argv[])
{
class Sphere ball1; // Create an instance of Sphere called ball1 (a cookie!)
class Sphere *ball2; // Create a pointer to store the location of an instance of Sphere
ball1.radius=10; // Make the radius of ball1 = 10
ball2=new Sphere(); // Create a new instance of a sphere and assign its location to ball2
ball2->radius=10; // Make ball2 radius equal to 10
(*ball2).radius=10; // This is the same as the line above, you can dereference a
// pointer and then use the dot to access members, (*p). same as ->
cout << ball1.radius << "," << ball2->radius; // Print value of radius in ball1 and ball2
int x; cin >> x; // Wait for keypress
}
```



Part 4: Creating a Windows Forms Application using C/C++

```
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{
label1->Text="Hello";
}
```

button1

Hello

```
this->button1->Text = L"Start";
this->button1->UseVisualStyleBackColor = true;
this->button1->Click += gcnew System::EventHandler(this, &Form1::button1_Click);
private: System::Void timer1_Tick(System::Object^ sender, System::EventArgs^ e)
{
count++;
label1->Text="Time:"+Convert::ToString((float)count/10.0f);
}
```

start

Time: 10.4

```
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{
//label1->Text="Hello";
timer1->Start();
}
private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e)
{
timer1->Stop();
}
```

stop start

Time: 10.5

To do: Use this code base to create your own stop watch application. Submit the Form1.h source code and a screen capture image of the application running; show your running code to a demonstrator before you leave.

```
#pragma once
```

```
#include <time.h>

namespace WF_App1 {
	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	int count = 0;
	clock_t start_time, current_time;

	public ref class Form1 : public System::Windows::Forms::Form {
	public:
		Form1(void) {
			InitializeComponent();
		}

	protected:
		~Form1() {
			if (components) {
				delete components;
			}
		}

	private:
		System::Windows::Forms::Button^ Start;
		System::Windows::Forms::Label^ label1;
		System::Windows::Forms::Timer^ timer1;
		System::Windows::Forms::Button^ button1;
		System::ComponentModel::IContainer^ components;

#pragma region Windows Form Designer generated code
		void InitializeComponent(void) {
			this->components = gcnew System::ComponentModel::Container();
			this->Start = gcnew System::Windows::Forms::Button();
			this->label1 = gcnew System::Windows::Forms::Label();
			this->timer1 = gcnew System::Windows::Forms::Timer(this->components);
			this->button1 = gcnew System::Windows::Forms::Button();

			// Start Button Setup
			this->Start->Location = System::Drawing::Point(155, 28);
			this->Start->Name = L"Start";
```

```
		this->Start->Size = System::Drawing::Size(112, 38);
		this->Start->Text = L"Start";
		this->Start->UseVisualStyleBackColor = true;
		this->Start->Click += gcnew System::EventHandler(this, &Form1::button1_Click);

		// Label Setup
		this->label1->AutoSize = true;
		this->label1->Font = gcnew System::Drawing::Font(L"Arial Narrow", 48,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(0));
		this->label1->Location = System::Drawing::Point(55, 73);
		this->label1->Name = L"label1";
		this->label1->Size = System::Drawing::Size(207, 147);
		this->label1->Text = L"0.0";

		// Timer Setup
		this->timer1->Tick += gcnew System::EventHandler(this, &Form1::timer1_Tick);

		// Stop Button Setup
		this->button1->Location = System::Drawing::Point(12, 32);
		this->button1->Name = L"button1";
		this->button1->Size = System::Drawing::Size(112, 38);
		this->button1->Text = L"Stop";
		this->button1->UseVisualStyleBackColor = true;
		this->button1->Click += gcnew System::EventHandler(this,
&Form1::button1_Click_1);

		// Form Setup
		this->AutoScaleDimensions = System::Drawing::SizeF(12, 24);
		this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
		this->ClientSize = System::Drawing::Size(274, 229);
		this->Controls->Add(this->button1);
		this->Controls->Add(this->label1);
		this->Controls->Add(this->Start);
		this->Name = L"Form1";
		this->Text = L"Form1";
		this->ResumeLayout(false);
		this->PerformLayout();
	}
#pragma endregion

	private:
	System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
		timer1->Start();
```

```
            start_time = clock();
        }

        System::Void timer1_Tick(System::Object^ sender, System::EventArgs^ e) {
            current_time = clock();
            label1->Text = "Time:" + String::Format("{0:0.00}", (float)(current_time -
start_time) / 1000.0f);
        }

        System::Void button1_Click_1(System::Object^ sender, System::EventArgs^ e) {
            timer1->Stop();
        }
    };
}
```

Time: 1.5