1. Consider the following Java bytecode sequence for a function:

```java
public static int example();
Code:
0: iconst_4
1: istore_1
2: iload_1
3: iinc 1, -1
6: iload_1
7: ifne 2
10: iadd
11: iadd
12: iadd
13: ireturn
```

(a) If this function were executed, what operations would it perform on the stack and what value would it return?

(b) This function will fail to pass the Java bytecode verifier. Why, exactly?

(c) Consider the following translation of the function into (not especially efficient) TALx86 assembly code:

```assembly
example: ∀ρ:Ts. {esp: sptr {eax: B4, esp:sptr ρ} :: ρ} 
mov ebx, 4
top: ∀ρ:Ts. {ebx: B4, esp: sptr ρ}
push ebx
dec ebx
cmp ebx, 0
jne tapp(top,<B4::ρ>)
pop eax
pop ebx
add eax, ebx
pop ebx
add eax, ebx
pop ebx
add eax, ebx
retn
```

(Note: Recall that the X86 uses two-address code; the convention for this assembler is that the destination register is listed first. So, for example, each add instruction adds ebx to eax and puts the result in eax.)

Is this TALx86 function well-typed? Explain why or why not.

2. CompCert

(a) Leroy defines what he means by a verified compiler. Explain why it is trivial to write such a compiler.

(b) Suppose you want to write a formal semantics for your favorite source language. How would you suggest validating that your semantics is correct?

**How to submit your homework.**
Submit this homework by email to apt@cs.pdx.edu with subject line “CS577 HW8.” Put your answers into a single *plain text* file with your name in it, and attach it to your mail.