1. Consider the following program which is written in a strongly and dynamically typed language. What is its output? Explain it.

```
def foo(i):
    print(i)
    j = 10
    i = j;
    print(i)
    i = "ceng212"
    print(i, j)

foo(5)
```

2. Consider the following program which is written in a strongly and dynamically typed language. What is its output? Explain it.

```
def foo(i):
    if i > 0:
        print(i+1)
    else:
        print("The parameter is: " + i)

def mymain():
    foo(-5)

mymain()
```

3. In the following code, a) which of the variables will a compiler consider to have compatible types under structural equivalence? b) Under strict name equivalence? c) Under loose name equivalence?

```
type T = array [1..10] of integer
      S = T
A : T
B : T
C : S
D : array [1..10] of integer
```

Solution:

```
5
10
ceng212 10
```

```
print("The parameter is: " + i)
TypeError: can only concatenate str (not "int") to str
```

a) All of them (A, B, C, D) are structurally equivalent – so compatible.
b) Under strict name equivalence, A and B are compatible to each other.
c) Under loose name equivalence, S is an alias to T, S can be equivalent to T. So, A, B and C are compatible to each other.