Problem 1

In [ ]:

```python
%%typecheck
import typing as tp

def foo(a: tp.Optional[int]) -> int:
    return a + 1

a: int = {"a": 1}.get("a")
foo(a)
```

Problem 2

In [ ]:

```python
%%typecheck
import typing as tp

aa = [1, "2", [3]]
aa[0] += 1
aa.append({1, 3})
```

Problem 3

In [ ]:

```python
%%typecheck
import typing as tp


def foo(a: tp.Set[int]) -> None:
    pass

a = {1.1, 2.1, 3.1}
foo(a)
b = {1, 2, 3}
foo(b)
c = {True, False}
foo(c)
```

Problem 4

In [ ]:

```
%%typecheck
import typing as tp


def foo(a: tp.AbstractSet[int]) -> None:
    pass

a = {1.1, 2.1, 3.1}
foo(a)
b = {1, 2, 3}
foo(b)
c = {True, False}
foo(c)
```

Problem 5

In [ ]:

```
%%typecheck
import typing as tp

class A:
    pass

class B(A):
    pass

def foo(a: A) -> B:
    return a.__class__()


foo(A())
foo(B())
```

Problem 6

In [ ]:

```
%%typecheck
class T:
    pass

class S(T):
    pass

class A:
    VAR = T()

class B(A):
    VAR = S()

class C(B):
    VAR = T()

```

Problem 7

```
In [ ]:
%%typecheck
import typing as tp

class A:
    pass

class B(A):
    pass

def g(f: tp.Callable[[A], B]) -> None:
    pass

def f1(a: A) -> A:
    pass

def f2(a: A) -> B:
    pass

def f3(a: B) -> A:
    pass

def f4(a: B) -> B:
    pass

g(f1)
g(f2)
g(f3)
g(f4)
```

Problem 8

In [ ]:

```python
%%typecheck
import typing as tp


def foo(a: tp.Iterable[str]) -> bool:
    b = len(a)
    c = sum(1 for i in a)
    return b == c

foo(["a", "b"])
foo("ab")
foo({"a": 2})

class A:
    def __len__(self) -> int:
        return 1

foo(A())

class B:
    def __iter__(self) -> tp.Iterator[int]:
        return iter([])

foo(B())


class C:
    def __iter__(self) -> tp.Iterator[str]:
        return iter([])

foo(C())
```

Problem 9

In [ ]:

```
%%typecheck
import typing as tp

class Fooable(tp.Protocol, tp.Sized):
    def foo(self, a: int) -> None:
        pass


class A(Fooable):
    def __len__(self) -> int:
        return 10

class B:
    def foo(self, a: int) -> None:
        pass

    def __len__(self) -> int:
        return 10

class C:
    def foo(self, a: int) -> None:
        pass


def foo(a: int) -> None:
    pass


def f(c: Fooable) -> None:
    c.foo(1)
    len(c)
    "10" in c

f([])
f(A())
f(B())
f(C())
f(foo)
```

Problem 10

In [ ]:

```python
%%typecheck
import typing as tp

T = tp.TypeVar("T", bound=tp.SupportsFloat, covariant=True)

class A(tp.Generic[T]):
    def __init__(self, a: T) -> None:
        self._a: tp.SupportsFloat = a

    def increment(self) -> float:
        self._a = float(self._a) + 1
        return self._a


A(1)
A(1.2)
A(True)
A("1.3")

class B:
    def __float__(self) -> float:
        return 1.1

A(B())

def g(a: A[int]) -> None:
    pass

g(A(1.4))
g(A(True))
```