7.

```
# Number 7 Multiple Choice Question
# The question has so many bugs and has lost or syntax errors.
# Original Code:

def main():
    myCount - Count()     # this is a typo, it should be "=" not "-"
    times = 0
for i in range(0, 100):   # this should be indented to the main function
    increment (myCount, times)
print(“myCount.count =”’ myCount.count, “times =”, times)
def increment(c, times)
    c.count += 1
    times += 1
class Count:
    def __init__(self):
        self.count = 0

main()


# I debugged the code and got this code

def main():
    myCount = Count()
    times = 0

    for i in range(0, 100):
        increment(myCount, times)

    print("myCount.count =", myCount.count, "times =", times)


def increment(c, times):
    c.count += 1
    times += 1


class Count:
    def __init__(self):
        self.count = 0


main()
```

Output

```
# Output:
# myCount.count = 100 times = 0
# So the answer to question 7 is "B"
```

16.

```
import math


class Point:
    def __init__(self, x, y) -> None:
        self.x = x
        self.y = y

    def __str__(self) -> str:
        return f"Center: ({self.x}, {self.y})"


class Circle(Point):
    def __init__(self, radius, x, y) -> None:
        super().__init__(x, y)
        self.radius = radius
        self.area = self.circle_area()

    def circle_area(self):
        return math.pi * self.radius * self.radius

    def __str__(self) -> str:
        return f"{super().__str__()}, Radius: {self.radius}"


class Cylinder(Circle):
    def __init__(self, height, radius, x, y) -> None:
        super().__init__(radius, x, y)
        self.height = height

    def surface_area(self):
        return 2 * math.pi * self.radius * self.height + 2 * math.pi *
self.radius * self.radius

    def volume(self):
        return self.circle_area() * self.height

    def __str__(self) -> str:
        return f"{super().__str__()}, Surface Area: {self.surface_area()},
Volume: {self.volume()}"


c = Circle(10, 5, 10)
```

```
print(c.circle_area())
print(c.__str__())

cy = Cylinder(100, 50, 10, 50)
print(cy.surface_area())
print(cy.volume())
print(cy.__str__())
```

Output

```
Center: (5, 10), Radius: 10
47123.8898038469
785398.1633974483
Center: (10, 50), Radius: 50, Surface Area: 47123.8898038469, Volume:
785398.1633974483
```

19.

```
def characters(string):
    """
    For each character in the string, if the character is in the
dictionary, add one to the value of that key, otherwise,
    add the character to the dictionary with a value of 1.
    """
    dictionary = dict()
    # Iterating through the string and adding the characters to the
dictionary.
    for i in string:
        keys = dictionary.keys()

        if i in keys:
            dictionary[i] += 1
        else:
            dictionary[i] = 1

    return dictionary


# Sorting the dictionary by the keys.
sorted_dict = dict(sorted(characters("William".lower()).items()))
print(sorted_dict)
```

Output

```
{'a': 1, 'i': 2, 'l': 2, 'm': 1, 'w': 1}
```

20.

```
def store_input() -> list:
    # Creating a list of dictionaries.
    students = list()
    # Asking the user to input a name and age 10 times.
    for i in range(10):
        name: str = input("Enter name: ")
        age: int = int(input("Enter age: "))
        students.append({"name": name, "age": age})

    return students


def check_eligibility(students: list) -> list:
    # Checking if the age is less than 16 and if it is, it is setting the
eligible key to False.
    for i in students:
        if i["age"] < 16:
            i["eligible"] = False
        else:
            i["eligible"] = True

    return students


def print_results(results: list) -> None:
    """
    It takes a list of dictionaries, and prints out the names and ages of
the people who are eligible to drive, and the names
    and ages of the people who are not eligible to vote
    """
    eligible = list()
    not_eligible = list()

    # Checking if the eligible key is True or False, and if it is True, it
is appending it to the eligible list,
    # and if it is False, it is appending it to the not_eligible list.
    for i in results:
        if i["eligible"]:
            eligible.append(i)
        else:
            not_eligible.append(i)

    # Checking if the eligible list is empty or not, and if it is not
empty, it is printing out the names and ages of
    # the people who are eligible to drive, and if the not_eligible list is
not empty, it is printing out the names
    # and ages of the people who are not eligible to drive.
    if len(eligible) != 0:
        print("Eligible")
```

```
        for i in eligible:
            print(i["name"], i["age"], end="\n\n")
    if len(not_eligible) != 0:
        print("Not Eligible")
        for i in not_eligible:
            print(i["name"], i["age"], end="\n\n")


def main():
    students = store_input()
    student_results = check_eligibility(students)
    print_results(student_results)


main()
```

Output

```
Enter name: William
Enter age: 19
Enter name: AK
Enter age: 18
Enter name: Jack
Enter age: 17
Enter name: Kyle
Enter age: 16
Enter name: Mikael
Enter age: 15
Enter name: Daphne
Enter age: 14
Enter name: Arkar
Enter age: 20
Enter name: Tayza
Enter age: 16
Enter name: Kylie
Enter age: 19
Enter name: Philip
Enter age: 21
Eligible
William 19

AK 18

Jack 17

Kyle 16

Arkar 20
```

```
Tayza 16

Kylie 19

Philip 21

Not Eligible
Mikael 15

Daphne 14
```