Rubric: IP EndSem Examination

The components of the rubric are binary. If the particular constraint and rationale is not satisfied, marks would not be awarded for that component.

Q1. a.
Expected Output: `[4, 0, 2, 2, 4, 4, 5]`

5 marks for correct output, 0 otherwise

Q1. b.
i.
1 marks if mentioned a = 2
1 mark if mentioned b = 4
ii.
3 marks if mentioned that there will 2 elements in z

Q2. a.
5 marks if correct code is written -

A sample solution -

```
def findstudents(records, cutoff):
    l = []
    for rollno in records:
        if records[rollno]["cgpa"] > cutoff:
            rec = (rollno, records[rollno]["Name"], records[rollno]["cgpa"] )
            l.append(rec)
    return l
```

3 marks if all accesses to dictionary are there and are proper (but the logic or the loop is wrong)

1 mark if at least 2 dictionary access are correctly shown.

Q2. b.
5 marks if correct 1 line list comprehension is written -

2 or 3 marks if partially correct (i.e. the first part of comprehension is fine but the condition is missing or wrong)

One of the ways to write the list comprehension-

```
c = [a[i]*b[i] for i in range(len(a)) if a[i]%2==0 and b[i]%2==0]
```

Q3.
0.5 mark for each correct print statement

Expected print statements:
32
3
7
23
63
2
45
9
1
2

Q4.

a. 2 marks for writing anything on the lines of that the zero division error will happen.
b. 3 marks for writing that the assertion error will be raised and then will be handled in the except block. And, 'Integer required' message will be printed. (Give 50% marks if only one of the two is mentioned)

Q5.

a. The expected output that will fetch 5 marks is : `[[[[0]]]]`
b. 2 marks for mentioning **No.**
   3 marks for reasoning that the base case of the recursion is not inclusive to all values of x, y and z for eg. for the cases where y>x. (Here, example is optional)

Q6

a) → 5 marks (full marks if function is correct - i.e. base case and recursive part both are correct, 2 marks if base case is correct + 1 or 2 marks if part of the logic of recursive part is correct)

```
def removeDup_r(myStr):
    if len(myStr)<2:
        return myStr

    if myStr[0]!=myStr[1]:
        return myStr[0] + removeDup_r(myStr[1:])

    return removeDup_r(myStr[1:])
```

b) → 5 marks (5 if fully correct, 3 if the initialization is correct and loop logic is partly correct, 1 - if some correct part is there (initialization or some part of loop)

```
def removeDup_w(myStr):
  s =''
  i=1
  while(i<len(myStr)):
    if myStr[i]!=myStr[i-1]:
      s +=myStr[i-1]
    if i == len(myStr)-1:
      s+=myStr[i]
    i+=1

  return s
```

In part a:

- Base case correctly mentioned →2 marks
- Logic is correct → 2 marks
- Return statement include → 1 mark

In part b:

- Loop correctly implemented using while → 2 marks
- Doesn't go in infinite loop → 1 mark
- Logic is correct and return statement included → 1 mark

Q7

a) → 5 marks

```
print(list(set(l1).union(l2)))
```

b) → 5 marks

```
print(set(d1.items()).intersection(set(d2.items())))
```

In both these parts, some students might not have used "print". They would have directly stated the line. **If logic is correct they should be given full marks.** Some students might have stated a bit different but logically correct and sound answer. They should be given full marks as well.

3 marks - if sets are used but the operation is wrong or there is some other mistakes

Q8

a) → 5 marks

```
fi = open('datain.txt','r')
y = fi.read()
print(y)
fi.close()
list_of_num = [int(num) for num in y.split()]
avg = sum(list_of_num)/ len(list_of_num)

fo = open('dataout.txt', 'w+')
fo.write(str(avg))
fo.close()
```

Some students might have opened files using "**with**". If not, file must be closed, and their order of closing could be different.
File opened and closed properly → 2 marks
Logic of reading the content and finding the average is correct → 2 marks
While writing out to the ''dataout.txt'', **avg** has been typecasted to **string** → 1 mark

b) → 5 marks

```
import myfns

x = int(input())
y = int(input())

a1 = myfns.sum_sq(x,y)
a2 = myfns.sum_cu(x,y)

## again passing a1 and a2 as in

a3 = myfns.sum_sq(a1,a2)
print(a3)
```

Import statement have been used → 1 marks [some student directly would have taken input in a single line, which is also correct]
Input taken and typecasted → 1 marks
Logic is correct, i.e, 2 calls to sum_sq and 1 call to sum_cu → 3 marks

Q9

a) → 5 marks

```
def addgrade(self, course, cr, gd):
    newTup = (course, cr, gd)
    self.grades.append(newTup)
```

b) → 5 marks

```
def cgpa(self):
    gradeTot, credTot = 0, 0
    for detail in self.grades:
      gradeTot += detail[1]*detail[2]
      credTot  +=detail[1]
    return gradeTot/credTot
```

c) → 5 marks

```
def __str__(self):
    return (f"Name: {self.name}, Rollno: {self.rollno}, CGPA: {self.cgpa()}")
```

5 marks of each of the parts in a, b and c. The variable names and numbers can vary. No marks if logic is not correct. If the use of self has been missed but the logic is correct, 1 mark will be deducted out of 5.

In (b), give 3 marks if basic logic is correct, but some mistake in operations or loop condition.