NEAR EAST UNIVERSITY

FACULTY OF ENGINEERING

2024-25 FALL SEMESTER

ECC102 – PROGRAMMING AND PROBLEM-SOLVING

MIDTERM EXAM

Name-Surname:	Std. No:
Department:	Signature:

- You have **100 minutes** to finish this exam.
- **WRITE CLEARLY.** We can't give you credit if we can't read and understand your answers!
- **Calculators, mobile phones, laptops, tablets, or other electronic devices are PROHIBITED.**
- A **single double-sided worksheet** including your **name and student number** is allowed.
- You have **three sections** in this exam.
- The first section includes determining the outputs of the given programs (8 questions).
- The second section includes **finding and correcting the mistakes** of the given programs (2 questions).
- The last section is the **code writing** section (3 questions).

SECTION I

Determine the outputs of the given Python codes (16 pts -- 2 pts. for each question). Write down your answers to the "Output" parts provided for each question.

```
for z in [2, 4, 7, 9]:
    print(z - 1)
```

Output:
1
3
6
8

```
print(2*'No' + 3*'!')
```

Output: NoNo!!!

```
def f1():
 print('Hi')
def f2():
 print('Lo')
f2()
f1()
f1()
```

Output:
Lo
Hi
Hi

```
def s(x):
 return x*x
tot = 0
for n in [1, 3, 5]:
 tot = tot + s(n)
print(tot)
```

Output: 35

```
def func(x):
 print(2*x)
func(5)
```

Output: 10

```
def func(x):
 print(2*x)
func('5')
```

Output: 55

```
str = "i love programming"
str = str.replace('love','<3')
print(str.capitalize())
```

Output: I <3 programming

```
str = "ecc_102"
print(str.isalnum())
```

Output: False

SECTION II

Determine the mistakes and correct the given Python codes.

1. The following program aims to create an empty dictionary and append key-value pairs. The **keys are between 1-5 (both included)**, and the **values are the squares of the key values as strings**. This is the expected output of the program: {1: '1', 2: '4', 3: '9', 4: '16', 5: '25'}.
However, the first-year student made a mistake that produced errors when the program was executed. Determine and correct the mistakes without adding or removing any statement to obtain the expected output (**10 pts**.).

```
our_dict = {}
for i in range(0,5):
    our_dict(i) = str(i**2)

print(our_dict)
```

Answer (corrected version):

```
our_dict = {}
for i in range(1,6):
    our_dict[i] = str(i**2)

print(our_dict)
```

2. The following program aims to **find the absolute value of the entered negative number**. The program would **display the absolute value of a negative number** and prompt, **"It is not a negative number...." if the entered value is not a negative**.

However, one of the students made a mistake that produced errors when the program was executed. Determine and correct the mistakes without adding or removing any statement **to execute the code without errors and to obtain the expected output**. You are free to modify the statements even though they have no syntax error. (**14 pts.**)

```
def absolute(n):
    flag, count = False, 1
    if n >= 0:
        flag = True
    else:
        for i in range(n,0):
            count = count + 1
    n = count
    return n

number = input("Enter an integer value:>")
a, b = absolute(number)
if a:
    print("It is not a negative number...")
else:
    print(f"The absolute value of {number} is {a}")
```

Answer (corrected version):

```
def absolute(n):
    flag, count = False, 0
    if n >= 0:
        flag = True
    else:
        for i in range(n,0):
            count = count + 1
    n = count
    return flag, n

number = int(input("Enter an integer value:>"))
a, b = absolute(number)
if a:
    print("It is not a negative number...")
else:
    print(f"The absolute value of {number} is {b}")
```

SECTION III

Write the Python programs for the given problems.

1. Write a Python program to **separate the fractional components** of the given **positive** value and **display them as an integer**. The program should display "**It is an integer**" if the given value **is an integer (20 pts.)**.

Example:

1. x = 3.141619, and the expected output is 141619.
2. x = 1.0, and the expected output is 0.
3. x = 5, and the expected output is "It is an integer."

Answer:

```
pi = 3.1416   #change pi value to check for different numbers...
pi = str(pi)
index = 0
flag = True
for i in range(len(pi)):

    if pi[i].isalnum():
        continue
    else:
        index = i+1
        flag = False

if flag:
    print("It is an integer")
else:
    new = int(pi[index:])
    print(new)
```

2. Write a Python program that partitions integers as being either less than or greater than a cutoff (threshold) value. Your program will **begin by reading the cutoff value from the user**. Then, it will **continue reading integers** from the user **until the user enters a negative number**. Your program must **include appropriate prompts and output messages** before the result. The output of your program will be all numbers entered by the user that are less than the cutoff value in the same order they were entered, followed by the cutoff value, followed by all of the integers greater than the cutoff value in the same order they were entered. **Example:** If the user started by entering a cutoff value of 50 and then entered the following integers:
10, 90, 80, 20, 40, 0, 60, -1 (one on each line), then your program should display the values 10, 20, 40, and 0 (the numbers less than the cutoff value), followed by 50 (the cutoff value), followed by the values 90, 80 and 60 (the numbers larger than the cutoff value). The output should be formatted so that **one value appears on each line**. (**20 pts**.)

Answer:

```
numbers = 0
cutoff = int(input("Enter a cutoff value:>"))
small = []
large = []
while numbers>=0:
    numbers = int(input("Enter a number (-ve to stop) :>"))
    if numbers < 0:
        break
    elif numbers <= cutoff:
        small.append(numbers)
    else:
        large.append(numbers)

print(*small, cutoff, *large)
```

3. Write a Python program to **check the similarity between two strings**. Your program should include a user-defined function "**similarity(s1,s2)**" to compare two strings in a function. Two strings will be passed to the function, and **your function will return two values: the number of matched characters at the same index and the similarity percentage (20 pts.)**.

USING BUILT-IN STRING FUNCTIONS ARE NOT ALLOWED!

Examples:

- If your str1=" ECC102" and str2=" ECC102", the program should print "Number of matched characters: 6" and "Similarity is 100%".
- If your str1=" ECC211" and str2=" ECC102", the program should print "Number of matched characters: 3" and "Similarity is 50%"
- If your str1=" ECC102" and str2=" MAT213", the program should print "Number of matched characters: 0" and "Similarity is 0%"

You must consider the length of the strings and focus on the longer ones if they have different lengths:

- If your str1=" ECC" and str2=" ECC102", the program should print "Number of matched characters: 3" and "Similarity is 50%".
- If your str1=" ECC" and str2=" MAT213", the program should print "Number of matched characters: 0" and "Similarity is 0%".

Use the next page to write your answer.

#Section 3 Question 3 answer.....

```
def similarity(s1,s2):
    if len(str1)<=len(str2):
        short = len(str1)
        long = len(str2)

    else:
        short = len(str2)
        long = len(str1)
    count = 0
    for i in range(short):
        if str1[i] == str2[i]:
            count = count + 1

    sim = (count * 100) / long

    return count, sim

str1 = "ECC102"
str2=  "MAT213"

n, s = similarity(str1,str2)

print(f"{n} characters are matched...")
print(f"Similarity is {s}%...")
```