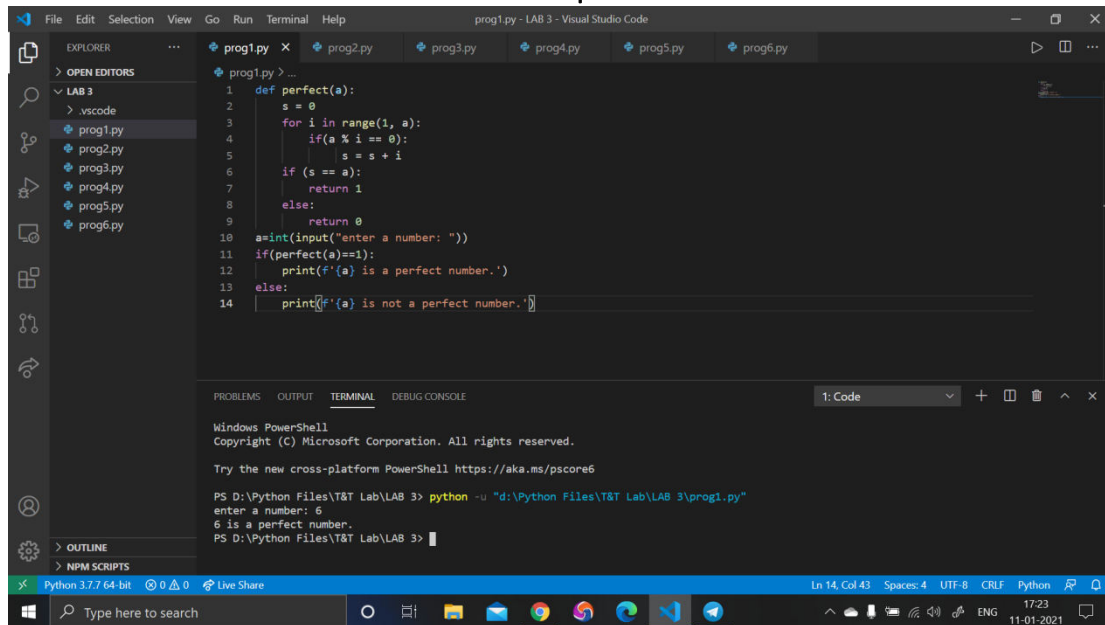


T&T LAB - 3

BISWARUP MUKHERJEE

ROLL - 1806468

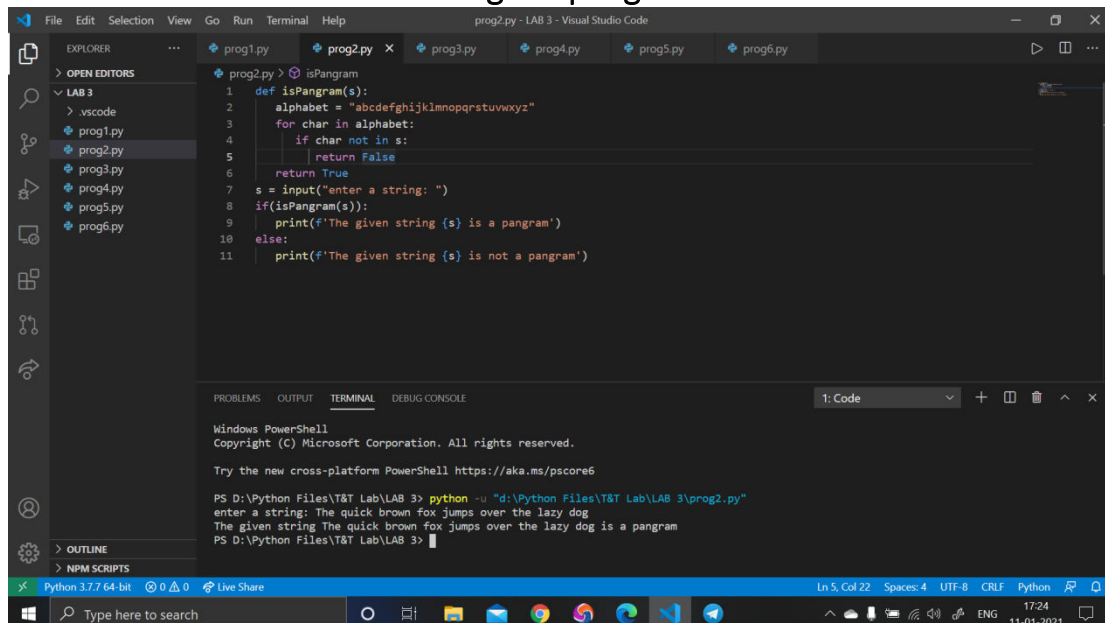
1. WAP to check whether a number is perfect or not.



```
1 def perfect(a):
2     s = 0
3     for i in range(1, a):
4         if(a % i == 0):
5             s = s + i
6     if (s == a):
7         return 1
8     else:
9         return 0
10 a=int(input("enter a number: "))
11 if(perfect(a)==1):
12     print(f'{a} is a perfect number.')
13 else:
14     print(f'{a} is not a perfect number.')
```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell <https://aka.ms/pscore6>
PS D:\Python Files\T&T Lab\LAB 3> python -u "d:\Python Files\T&T Lab\LAB 3\prog1.py"
enter a number: 6
6 is a perfect number.
PS D:\Python Files\T&T Lab\LAB 3>

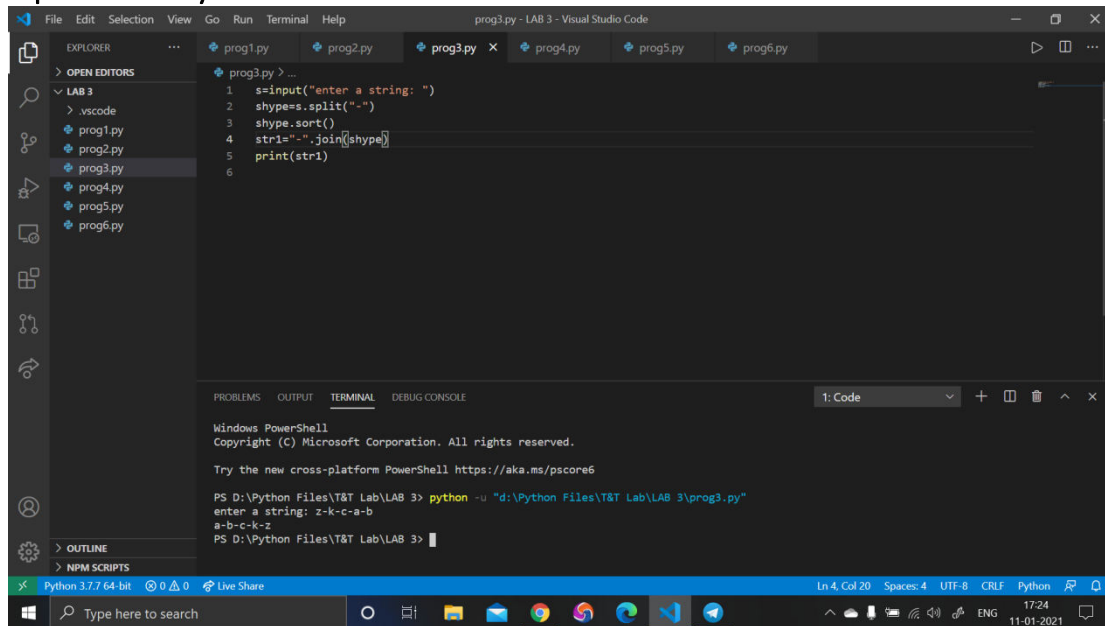
2. WAP to check whether a string is a pangram or not.



```
1 def isPangram(s):
2     alphabet = "abcdefghijklmnopqrstuvwxyz"
3     for char in alphabet:
4         if char not in s:
5             return False
6     return True
7 s = input("enter a string: ")
8 if(isPangram(s)):
9     print(f'The given string {s} is a pangram')
10 else:
11     print(f'The given string {s} is not a pangram')
```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell <https://aka.ms/pscore6>
PS D:\Python Files\T&T Lab\LAB 3> python -u "d:\Python Files\T&T Lab\LAB 3\prog2.py"
enter a string: The quick brown fox jumps over the lazy dog
The given string The quick brown fox jumps over the lazy dog is a pangram
PS D:\Python Files\T&T Lab\LAB 3>

3. WAP that accepts a hyphen - seperated sequence of words as input and prints the words in a hyphen - seperated sequence after sorting them alphabetically.



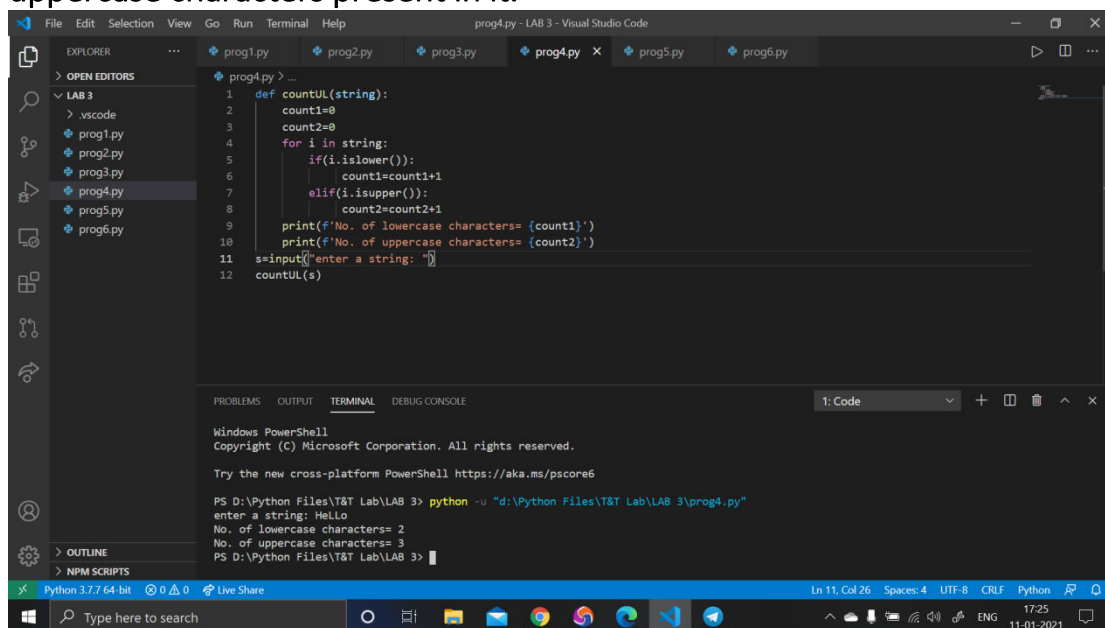
The screenshot shows the Visual Studio Code interface with a Python file named `prog3.py` open. The code in the editor is as follows:

```
1 s=input("enter a string: ")
2 shype=s.split("-")
3 shype.sort()
4 str1="-".join(shype)
5 print(str1)
6
```

The Explorer sidebar on the left shows a project named "LAB 3" containing several Python files, with `prog3.py` selected. The Terminal at the bottom shows the execution of the program:

```
PS D:\Python Files\T&T Lab\LAB 3> python -u "d:\Python Files\T&T Lab\LAB 3\prog3.py"
enter a string: z-k-c-a-b
a-b-c-k-z
PS D:\Python Files\T&T Lab\LAB 3>
```

4. WAP that accepts a string and counts the number of lowercase and uppercase characters present in it.



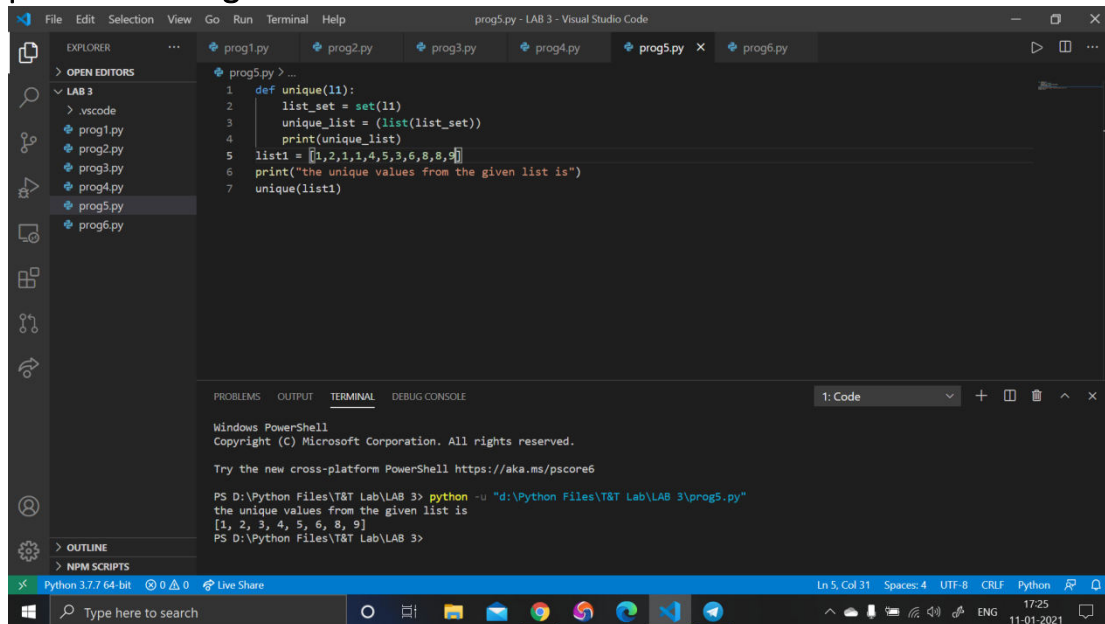
The screenshot shows the Visual Studio Code interface with a Python file named `prog4.py` open. The code in the editor is as follows:

```
1 def countUL(string):
2     count1=0
3     count2=0
4     for i in string:
5         if(i.islower()):
6             count1=count1+1
7         elif(i.isupper()):
8             count2=count2+1
9     print(f'No. of lowercase characters= {count1}')
10    print(f'No. of uppercase characters= {count2}')
11 s=input("enter a string: ")
12 countUL(s)
```

The Explorer sidebar on the left shows a project named "LAB 3" containing several Python files, with `prog4.py` selected. The Terminal at the bottom shows the execution of the program:

```
PS D:\Python Files\T&T Lab\LAB 3> python -u "d:\Python Files\T&T Lab\LAB 3\prog4.py"
enter a string: HeLlo
No. of lowercase characters= 2
No. of uppercase characters= 3
PS D:\Python Files\T&T Lab\LAB 3>
```

5. WAP that takes a list and returns a new list having the unique elements present in the given list.



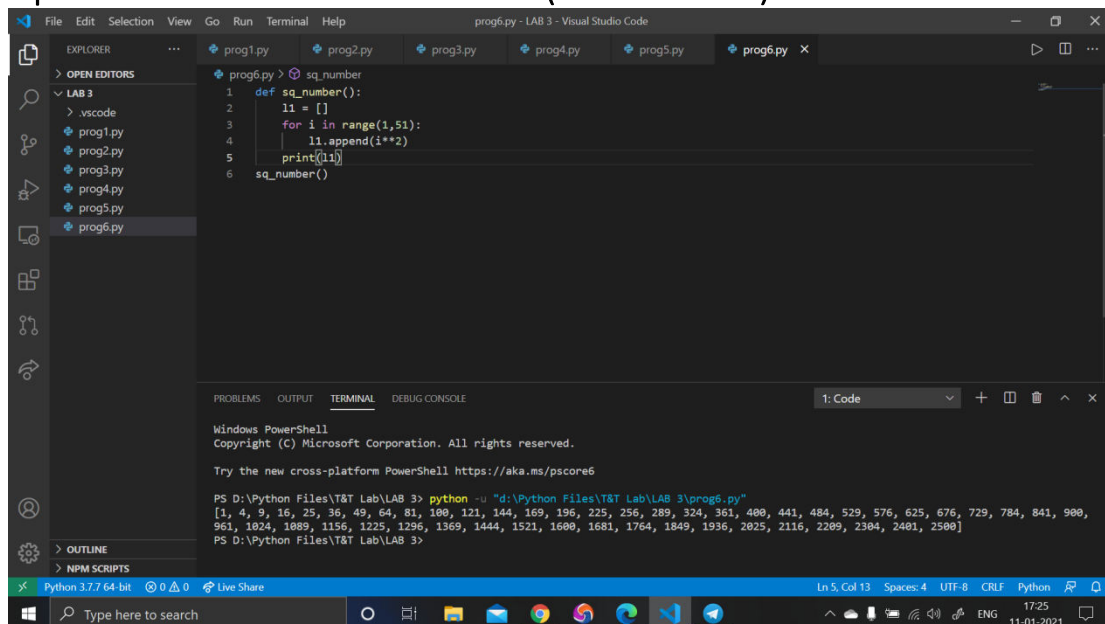
The screenshot shows the Visual Studio Code interface with a Python file named prog5.py. The code defines a function unique(l1) that uses a set to find unique elements in a list. The list l1 is [1, 2, 1, 1, 4, 5, 3, 6, 8, 8, 9]. The output in the terminal shows the unique values: [1, 2, 3, 4, 5, 6, 8, 9].

```
1 def unique(l1):
2     list_set = set(l1)
3     unique_list = (list(list_set))
4     print(unique_list)
5 list1 = [1, 2, 1, 1, 4, 5, 3, 6, 8, 8, 9]
6 print("the unique values from the given list is")
7 unique(list1)
```

Terminal Output:

```
PS D:\Python Files\T&T Lab\LAB 3> python -u "d:\Python Files\T&T Lab\LAB 3\prog5.py"
the unique values from the given list is
[1, 2, 3, 4, 5, 6, 8, 9]
PS D:\Python Files\T&T Lab\LAB 3>
```

6. WAP to print and create a list of integers where the values are the square of numbers between 1 to 50 (both included).



The screenshot shows the Visual Studio Code interface with a Python file named prog6.py. The code defines a function sq_number() that creates a list of squares from 1 to 50. The output in the terminal shows the list of squares: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400, 441, 484, 529, 576, 625, 676, 729, 784, 841, 900, 961, 1024, 1089, 1156, 1225, 1296, 1369, 1444, 1521, 1600, 1681, 1764, 1849, 1936, 2025, 2116, 2209, 2304, 2401, 2500].

```
1 def sq_number():
2     l1 = []
3     for i in range(1,51):
4         l1.append(i**2)
5     print(l1)
6 sq_number()
```

Terminal Output:

```
PS D:\Python Files\T&T Lab\LAB 3> python -u "d:\Python Files\T&T Lab\LAB 3\prog6.py"
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400, 441, 484, 529, 576, 625, 676, 729, 784, 841, 900, 961, 1024, 1089, 1156, 1225, 1296, 1369, 1444, 1521, 1600, 1681, 1764, 1849, 1936, 2025, 2116, 2209, 2304, 2401, 2500]
PS D:\Python Files\T&T Lab\LAB 3>
```