ASSIGNMENT 3

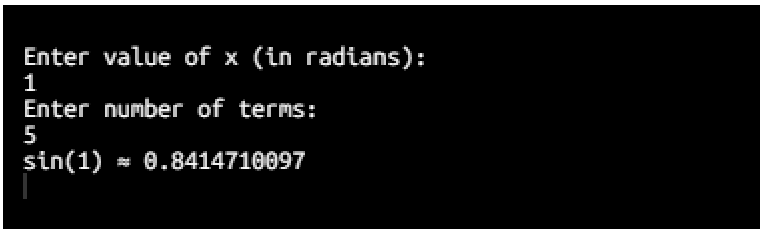
1) Write a shell program to print the sum of the following series:

a. $\sin(x) = x - x^3 / 3! + x^5 / 5! - x^7 / 7! + \ldots \pm x^n / n!$

CODE:

```
#!/bin/bash


echo "Enter value of x (in radians):"

read x

echo "Enter number of terms:"

read n


# We use awk to handle the floating point math

result=$(awk -v x="$x" -v n="$n" 'BEGIN {

   sum = x;

   term = x;

   for (k = 1; k < n; k++) {

      term = term * -1 * (x * x) / ((2 * k) * (2 * k + 1));

      sum = sum + term;

   }

   printf "%.10f", sum

}')


echo "sin($x) ≈ $result"
```

OUTPUT:



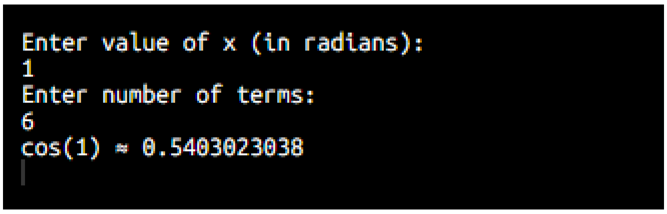
```
Enter value of x (in radians):
1
Enter number of terms:
5
sin(1) ≈ 0.8414710097
```

b. $\cos(x) = 1 - x^2 / 2! + x^4 / 4! - x^6 / 6! + \ldots \pm x^n / n!$

CODE:

```
#!/bin/bash


echo "Enter value of x (in radians):"
read x
echo "Enter number of terms:"
read n


# We use awk because it handles floating-point math and large numbers natively
awk -v x="$x" -v n="$n" 'BEGIN {
   sum = 1;    # The first term of cos(x) is 1
   term = 1;


   # Loop from 1 to n-1 to calculate subsequent terms
   for (i = 1; i < n; i++) {
       # Recurrence: Term_i = Term_{i-1} * (-x^2) / ((2*i-1)*(2*i))
       term = term * -1 * (x * x) / ((2 * i - 1) * (2 * i));
       sum = sum + term;
   }


   printf "cos(%g) ≈ %.10f\n", x, sum
}'
```

OUTPUT:



```
Enter value of x (in radians):
1
Enter number of terms:
6
cos(1) ≈ 0.5403023038
```

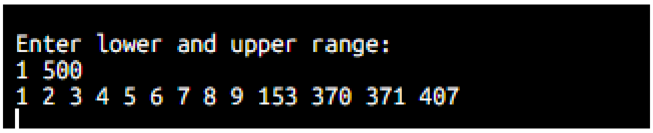
2) Write a shell program to display Armstrong numbers within a user-defined range.

CODE:

```
#!/bin/bash
echo "Enter lower and upper range:"
read low high
```

```
for ((i=low; i<=high; i++)); do
   n=${#i}
   sum=0
   temp=$i
   while [ $temp -gt 0 ]; do
      digit=$((temp % 10))
      sum=$((sum + digit**n))
      temp=$((temp / 10))
   done
   if [ $sum -eq $i ]; then echo -n "$i "; fi
done
echo ""
```

OUTPUT:



```
Enter lower and upper range:
1 500
1 2 3 4 5 6 7 8 9 153 370 371 407
```

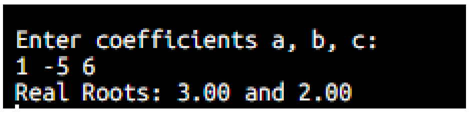
3) Write a shell program that takes coefficients of a quadratic equation as input and calculates its roots. Handle cases of real and complex roots. [If the discriminant D = b 2 – 4ac is negative, D will yield an imaginary number. For example, the quadratic equation x2 – 2x + 10 = 0, has a = 1, b = −2, c = 10. Therefore, D = √−36 = 6i and root α = (−b + √D) / 2a = 1 + 3i and root β = (−b – √D) / 2a = 1 – 3i.]

CODE:

```
#!/bin/bash
echo "Enter coefficients a, b, c:"
read a b c

python3 -c "
import math
a, b, c = $a, $b, $c
d = b**2 - 4*a*c
```

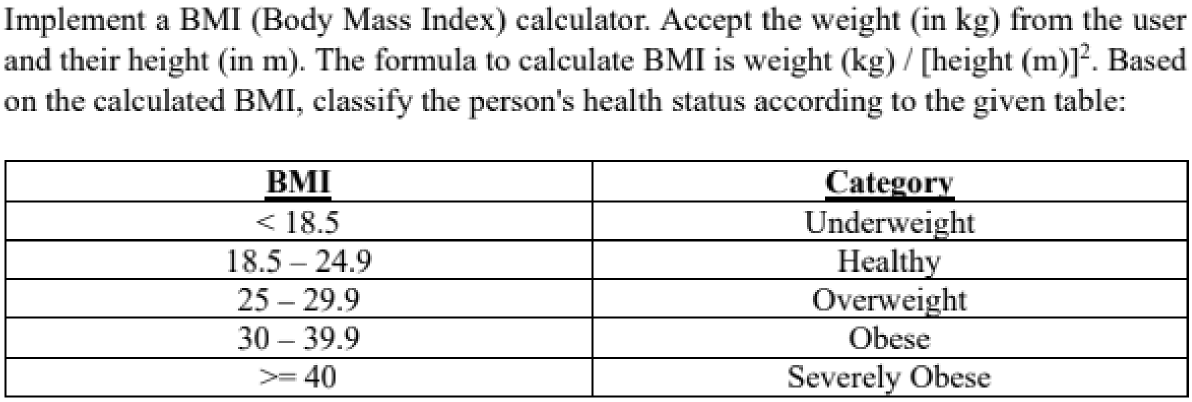
```
if d >= 0:
    r1 = (-b + math.sqrt(d)) / (2*a)
    r2 = (-b - math.sqrt(d)) / (2*a)
    print(f'Real Roots: {r1:.2f} and {r2:.2f}')
else:
    real = -b / (2*a)
    imag = math.sqrt(-d) / (2*a)
    print(f'Complex Roots: {real:.2f} + {imag:.2f}i and {real:.2f} - {imag:.2f}i')
"
```

OUTPUT:



```
Enter coefficients a, b, c:
1 -5 6
Real Roots: 3.00 and 2.00
```

4)



Implement a BMI (Body Mass Index) calculator. Accept the weight (in kg) from the user and their height (in m). The formula to calculate BMI is weight (kg) / [height (m)]^2. Based on the calculated BMI, classify the person's health status according to the given table:

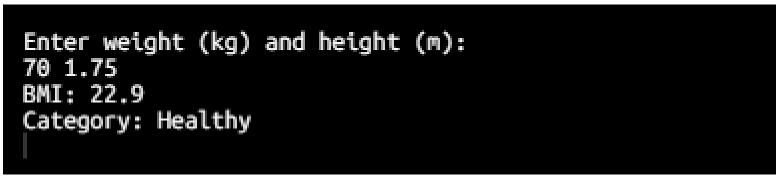
BMI	**Category**
< 18.5	Underweight
18.5 – 24.9	Healthy
25 – 29.9	Overweight
30 – 39.9	Obese
>= 40	Severely Obese

CODE:

```
#!/bin/bash
echo "Enter weight (kg) and height (m):"
read w h


bmi=$(awk -v w="$w" -v h="$h" 'BEGIN {printf "%.1f", w/(h*h)}')
echo "BMI: $bmi"


if (( $(echo "$bmi < 18.5" | awk '{print ($1 < 18.5)}') )); then
    echo "Category: Underweight"
```

```
elif (( $(echo "$bmi < 25" | awk '{print ($1 < 25)}') )); then
   echo "Category: Healthy"
elif (( $(echo "$bmi < 30" | awk '{print ($1 < 30)}') )); then
   echo "Category: Overweight"
elif (( $(echo "$bmi < 40" | awk '{print ($1 < 40)}') )); then
   echo "Category: Obese"
else
   echo "Category: Severely Obese"
fi
```

OUTPUT:



```
Enter weight (kg) and height (m):
70 1.75
BMI: 22.9
Category: Healthy
```

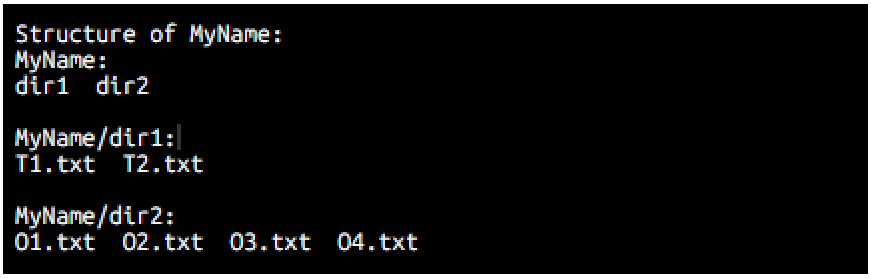
5) Create a directory with your name. Inside the directory create two sub-directories dir1 and dir2 respectively. Show the list of files and directories. Create two text files T1.txt and T2.txt inside dir1. Write ten email addresses corresponding to four different domain names (@gmail.com, @yahoo.com, @rediff.com, @teamfuture.in). Segregate the email addresses with respect to their domain name and save the email addresses inside four different text files O1.txt, O2.txt, O3.txt and O4.txt inside dir2.

CODE:

```
#!/bin/bash

# 1. Create directory with your name (replace 'MyName' with your actual name)
dir_name="MyName"
mkdir -p "$dir_name/dir1" "$dir_name/dir2"

# 2. Create the input file T1.txt inside dir1 with 10 email addresses
cat <<EOF > "$dir_name/dir1/T1.txt"
student1@gmail.com
student2@yahoo.com
```

```
student3@rediff.com
student4@teamfuture.in
student5@gmail.com
student6@yahoo.com
student7@rediff.com
student8@teamfuture.in
student9@gmail.com
student10@gmail.com
EOF

# 3. Create a dummy T2.txt just to satisfy the requirement of "two text files"
touch "$dir_name/dir1/T2.txt"

# 4. Segregate emails from T1.txt into dir2 based on domain
grep "@gmail.com" "$dir_name/dir1/T1.txt" > "$dir_name/dir2/O1.txt"
grep "@yahoo.com" "$dir_name/dir1/T1.txt" > "$dir_name/dir2/O2.txt"
grep "@rediff.com" "$dir_name/dir1/T1.txt" > "$dir_name/dir2/O3.txt"
grep "@teamfuture.in" "$dir_name/dir1/T1.txt" > "$dir_name/dir2/O4.txt"

# 5. Show the list of files and directories to verify
echo "Structure of $dir_name:"
ls -R "$dir_name"
```

OUTPUT:



```
Structure of MyName:
MyName:
dir1  dir2

MyName/dir1:
T1.txt  T2.txt

MyName/dir2:
O1.txt  O2.txt  O3.txt  O4.txt
```

6) Create three text files T1.txt containing the roll numbers and names of 10 students, T2.txt containing the names and heights of 10 students and T3.txt containing the roll numbers and grades of these same students. Find the highest grade, lowest grade, average grade and

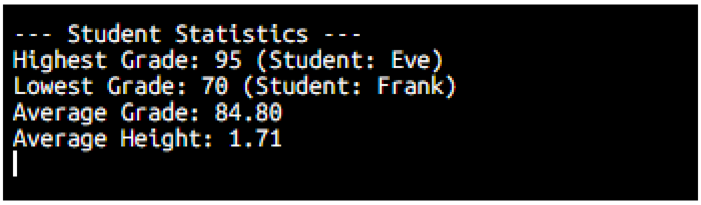
average height from the given data, and also find the corresponding student name(s). Store this output in a new text file O.txt.

CODE:

```
#!/bin/bash

# --- STEP 1: Create the Input Files ---
# These are the files the program will "read"
cat <<EOF > T1.txt
101 Alice
102 Bob
103 Charlie
104 David
105 Eve
106 Frank
107 Grace
108 Heidi
109 Ivan
110 Judy
EOF

cat <<EOF > T2.txt
Alice 1.65
Bob 1.80
Charlie 1.75
David 1.70
Eve 1.60
Frank 1.85
Grace 1.68
Heidi 1.72
Ivan 1.78
Judy 1.62
```

```
EOF

cat <<EOF > T3.txt
101 85
102 92
103 78
104 88
105 95
106 70
107 82
108 90
109 81
110 87
EOF

# --- STEP 2: The Logic ---
# We use AWK to link Roll Numbers to Names and calculate stats
awk '
  # If reading T1, store Roll Number -> Name
  FILENAME == "T1.txt" { name[$1] = $2 }

  # If reading T3, link Roll Number to Name and check grades
  FILENAME == "T3.txt" {
    student_name = name[$1];
    sumG += $2;
    count++;
    if (maxG == "" || $2 > maxG) { maxG = $2; maxN = student_name }
    if (minG == "" || $2 < minG) { minG = $2; minN = student_name }
  }

  # If reading T2, calculate average height
  FILENAME == "T2.txt" { sumH += $2 }


  END {
    printf "--- Student Statistics ---\n" > "O.txt"
    printf "Highest Grade: %d (Student: %s)\n", maxG, maxN >> "O.txt"
    printf "Lowest Grade: %d (Student: %s)\n", minG, minN >> "O.txt"
    printf "Average Grade: %.2f\n", sumG/count >> "O.txt"
    printf "Average Height: %.2f\n", sumH/count >> "O.txt"
  }
' T1.txt T3.txt T2.txt


# --- STEP 3: Show result ---
cat O.txt
```

OUTPUT:



```
--- Student Statistics ---
Highest Grade: 95 (Student: Eve)
Lowest Grade: 70 (Student: Frank)
Average Grade: 84.80
Average Height: 1.71
```

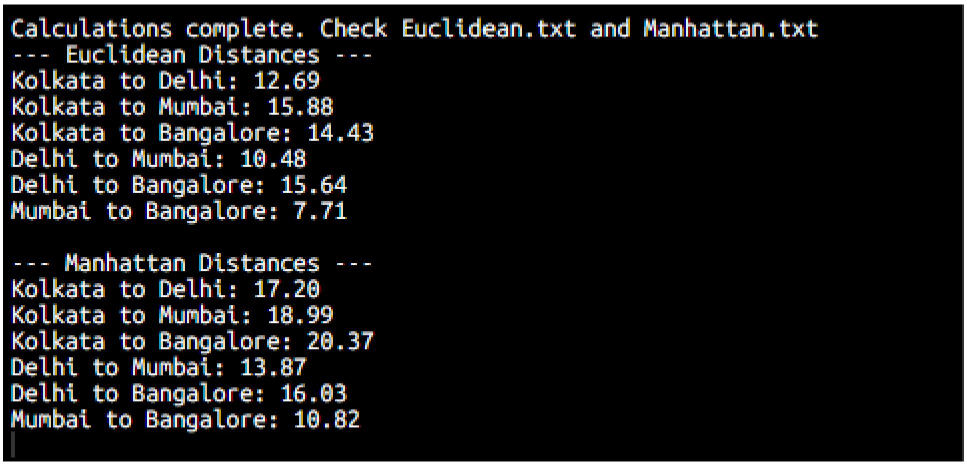
7) Calculate the Euclidean distance and Manhattan distance in between the cities. Coordinate information of the cities is provided in a file named City.csv. Euclidean and Manhattan distance between the cities will be calculated and stored in Euclidean.txt and Manhattan.txt respectively. [Coordinate information format for City.csv: city_name, longitude, latitude; Information format for Euclidean.txt and Manhattan.txt: source_city, destination_city, distance]

CODE:

```
#!/bin/bash


# --- STEP 1: Create the Input File (City.csv) ---
cat <<EOF > City.csv
city_name,longitude,latitude
Kolkata,88.36,22.57
```

```
Delhi,77.20,28.61
Mumbai,72.87,19.07
Bangalore,77.59,12.97
EOF

# --- STEP 2: Process with AWK ---
awk -F',' '
    # Skip the header line
    NR == 1 { next }

    # Store city data into arrays
    {
        name[NR] = $1;
        lon[NR] = $2;
        lat[NR] = $3;
        count++
    }

    END {
        # Loop through every pair of cities
        for (i = 2; i <= count + 1; i++) {
            for (j = i + 1; j <= count + 1; j++) {

                # Math calculations
                dx = lon[i] - lon[j]
                dy = lat[i] - lat[j]

                # Euclidean Distance: sqrt(dx^2 + dy^2)
                euc = sqrt(dx*dx + dy*dy)

                # Manhattan Distance: |dx| + |dy|
        abs_dx = (dx < 0 ? -dx : dx)
        abs_dy = (dy < 0 ? -dy : dy)
        man = abs_dx + abs_dy

        # Save to files
        printf "%s to %s: %.2f\n", name[i], name[j], euc >> "Euclidean.txt"
        printf "%s to %s: %.2f\n", name[i], name[j], man >> "Manhattan.txt"
      }
    }
    print "Calculations complete. Check Euclidean.txt and Manhattan.txt"
  }
' City.csv

# --- STEP 3: Show the Results ---
echo "--- Euclidean Distances ---"
cat Euclidean.txt
echo -e "\n--- Manhattan Distances ---"
cat Manhattan.txt
```

OUTPUT:



```
Calculations complete. Check Euclidean.txt and Manhattan.txt
--- Euclidean Distances ---
Kolkata to Delhi: 12.69
Kolkata to Mumbai: 15.88
Kolkata to Bangalore: 14.43
Delhi to Mumbai: 10.48
Delhi to Bangalore: 15.64
Mumbai to Bangalore: 7.71

--- Manhattan Distances ---
Kolkata to Delhi: 17.20
Kolkata to Mumbai: 18.99
Kolkata to Bangalore: 20.37
Delhi to Mumbai: 13.87
Delhi to Bangalore: 16.03
Mumbai to Bangalore: 10.82
```