Lab 3: Functions, File Processing, and Arrays

Name: ____________________________ University Number: ______________

Exercise 1: Recursive Function to Evaluate a Finite Sum

AIM:

An approximation to the function $x/(1-x)^2$ for $|x| < 1$ is given by the finite sum:

$$\frac{x}{(1-x)^2} = \sum_{i=1}^{n} ix^i$$

where n is a finite large number. Write a Python program that implements the recursive function `fsum(x, n)` to compute the above finite sum. Your program should also contain the code that displays this sum to 8 decimal places for x = 0.1, 0.2, 0.3, 0.4 and n = 2, 5, 10, 50, 100 by using this function.

ALGORITHM:

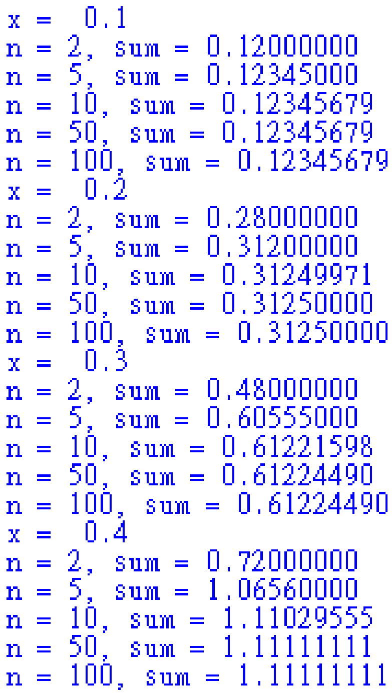
```
1. Start
2. Define the function fsum(x, n) as follows:
   (a) If n = 1, return x
   (b) Else, return n*x**n + fsum(x, n-1)
3. Display the results for calling the function with different values of x and n
4. End
```

PROGRAM:

```python
# Exercise 1: Recursive Approach to Evaluate a Finite Sum
# Written by F K Chow, HKU
# Latest Update: 2022/2/10

def fsum(x, n):
    """ Compute the finite sum for x/(1-x)^2 with a recursive app """
    if n == 1:
        return x
    return n*x**n + fsum(x, n-1)
```

```
# Display the results for calling the function with different values
# of x and n
for x in [0.1, 0.2, 0.3, 0.4]:
    print('x = ', x)
    for n in [2, 5, 10, 50, 100]:
        print('n = {:d}, sum = {:.8f}'.format(n, fsum(x, n)))
```

OUTPUT:



```
x =  0.1
n = 2, sum = 0.12000000
n = 5, sum = 0.12345000
n = 10, sum = 0.12345679
n = 50, sum = 0.12345679
n = 100, sum = 0.12345679
x =  0.2
n = 2, sum = 0.28000000
n = 5, sum = 0.31200000
n = 10, sum = 0.31249971
n = 50, sum = 0.31250000
n = 100, sum = 0.31250000
x =  0.3
n = 2, sum = 0.48000000
n = 5, sum = 0.60555000
n = 10, sum = 0.61221598
n = 50, sum = 0.61224490
n = 100, sum = 0.61224490
x =  0.4
n = 2, sum = 0.72000000
n = 5, sum = 1.06560000
n = 10, sum = 1.11029555
n = 50, sum = 1.11111111
n = 100, sum = 1.11111111
```

Exercise 2: Manipulating the Data from a Text File

AIM:

A text file called `HKFM.txt` contains the data of first marriages registered in Hong Kong from 1995 to 2020 (Source: https://www.censtatd.gov.hk/tc/scode160.html by Census and Statistics Department, HKSAR) which are delimited by tab as follows:

```
[Number of first marriages registered in HK by sex and age group]
Sex/Age group (years)    1995 2000 2005 2010 2015 2020
Male
16-19     289  232  220  175  155  31
20-24     4331 3076 3512 3737 3036 851
                        ⋮
>= 50     386  493  1690 1068 1014 581
Female
16-19     1213 966  935  683  511  98
20-24     10066     6613 7972 8286 5805 1545
                        ⋮
>= 50     140  93   156  227  320  319
```

Write a Python program that reads the data from this file, find the total number and dominant age group of first marriages registered in Hong Kong by sex and year, and finally print a table of the results with the following format on the screen:

```
   Sex  Year  Total Number  Dominant Age Group
----------------------------------------------
  Male  1995         34080               25-29
  Male  2000         26176               25-29
                        ⋮
Female  1995         34232               25-29
Female  2000         26605               25-29
                        ⋮
```

ALGORITHM:

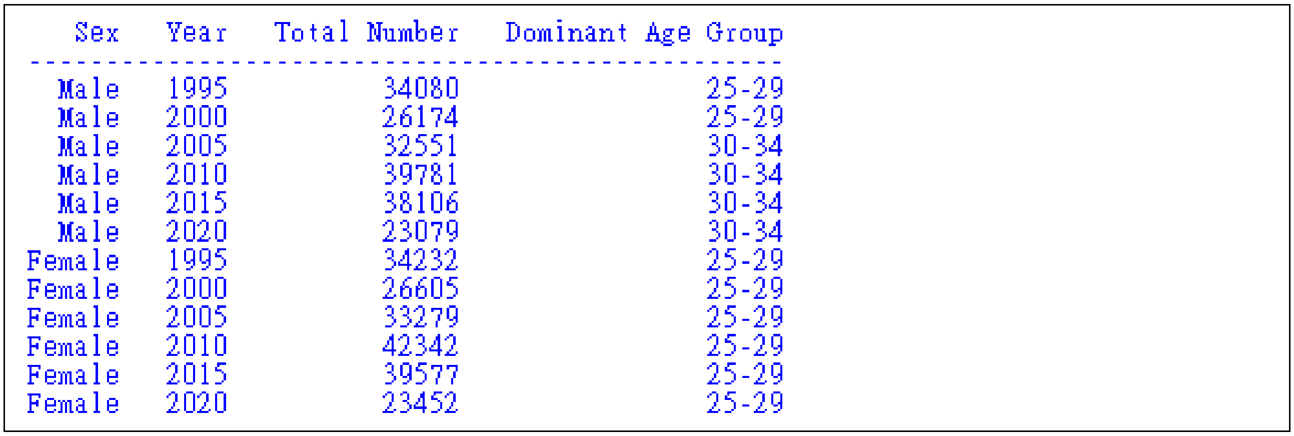
1. Start
2. Read the data of first marriages registered in HK from the file `HKFM.txt`

3. Find the total number and dominant age group of first marriages by sex and year
4. Print a table of the results on the screen
5. End

PROGRAM:

```
# Exercise 2: Manipulating the Data from a Text File
# Written by F K Chow, HKU
# Latest Update: 2022/2/10

# Read the data of first marriages registered in HK from the file
with open('HKFM.txt', 'r') as infile:
    line = infile.readline()    # Discard the title line
    line = infile.readline()
    year = line.strip('\n').split('\t')[1:]
    data = []
    sex = []
    for line in infile:
        if '\t' in line:    # Check if the line has tab-delimited data
            row = []
            for i in line.strip('\n').split('\t'):
                row.append(i)
            data.append(row)
        else:
            sex.append(line)

def getdata(year, rawdata):
    """ Find the total number and dominant age group """
    nyr = len(year)
    total = [0 for i in range(nyr)]
    dag = ['' for i in range(nyr)]
    for i in range(nyr):
        max = 0
        for j in range(len(rawdata)):
            num = int(rawdata[j][i+1])
            if max < num:
                max = num
                dag[i] = rawdata[j][0]
            total[i] += num
    return total, dag

# Find the total number and dominant age group of first marriages
# by sex and year
nyr = len(year)
nag = int(len(data)/2)  # Number of age groups
sex = ['Male', 'Female']
totalm, dagm = getdata(year, data[:nag])
totalf, dagf = getdata(year, data[nag:])

# Print a table of the results
print('{:>6s} {:>6s} {:>14s} {:>20s}'.format('Sex', 'Year',
                                             'Total Number',
                                             'Dominant Age Group'))
print('-'*49)
for i in range(nyr):
    print('{:>6s} {:>6s} {:>14d} {:>20s}'.format(sex[0], year[i],
                                                  totalm[i], dagm[i]))
for i in range(nyr):
    print('{:>6s} {:>6s} {:>14d} {:>20s}'.format(sex[1], year[i],
                                                  totalf[i], dagf[i]))
```

OUTPUT:



```
   Sex   Year   Total Number   Dominant Age Group
-------------------------------------------------
  Male   1995          34080                25-29
  Male   2000          26174                25-29
  Male   2005          32551                30-34
  Male   2010          39781                30-34
  Male   2015          38106                30-34
  Male   2020          23079                30-34
Female   1995          34232                25-29
Female   2000          26605                25-29
Female   2005          33279                25-29
Female   2010          42342                25-29
Female   2015          39577                25-29
Female   2020          23452                25-29
```

Exercise 3: Printing a Histogram to a Text File

AIM:

A text file called `HKPop2020data.txt` contains the data of the mid-year population in Hong Kong for 2020 (Source: Hong Kong Digest of Statistics 2021 Edition by Census and Statistics Department Hong Kong Special Administrative Region) which are delimited by space as follows:

```
[Mid-year population in Hong Kong by age group and sex for 2020]
[Age Group] [Male] [Female]
0-4 137400 127400
5-9 154600 144800
10-14 154400 150700
            ⋮
80-84 79900 94000
>=85 79100 138700
```

Write a Python program to show the statistics of the data in this file by printing a histogram in the following format to a text file `HKPop2020hist.txt`:

```
Mid-year Population in Hong Kong by Age Group and Sex for 2020
(in nearest ten thousands)
  0-4 | ##############&&&&&&&&&&&&& (14/13)
  5-9 | ###############&&&&&&&&&&&&&& (15/14)
10-14 | ###############&&&&&&&&&&&&&&& (15/15)
                      ⋮
 >=85 | ########&&&&&&&&&&&&&& (8/14)
#-Male; &-Female
```

ALGORITHM:

1. Start
2. Read the data of the 2020 mid-year population in HK from the file `HKPop2020data.txt`
3. Extract the statistics from the data and show them by printing a histogram to the file `HKPop2020hist.txt`
4. End

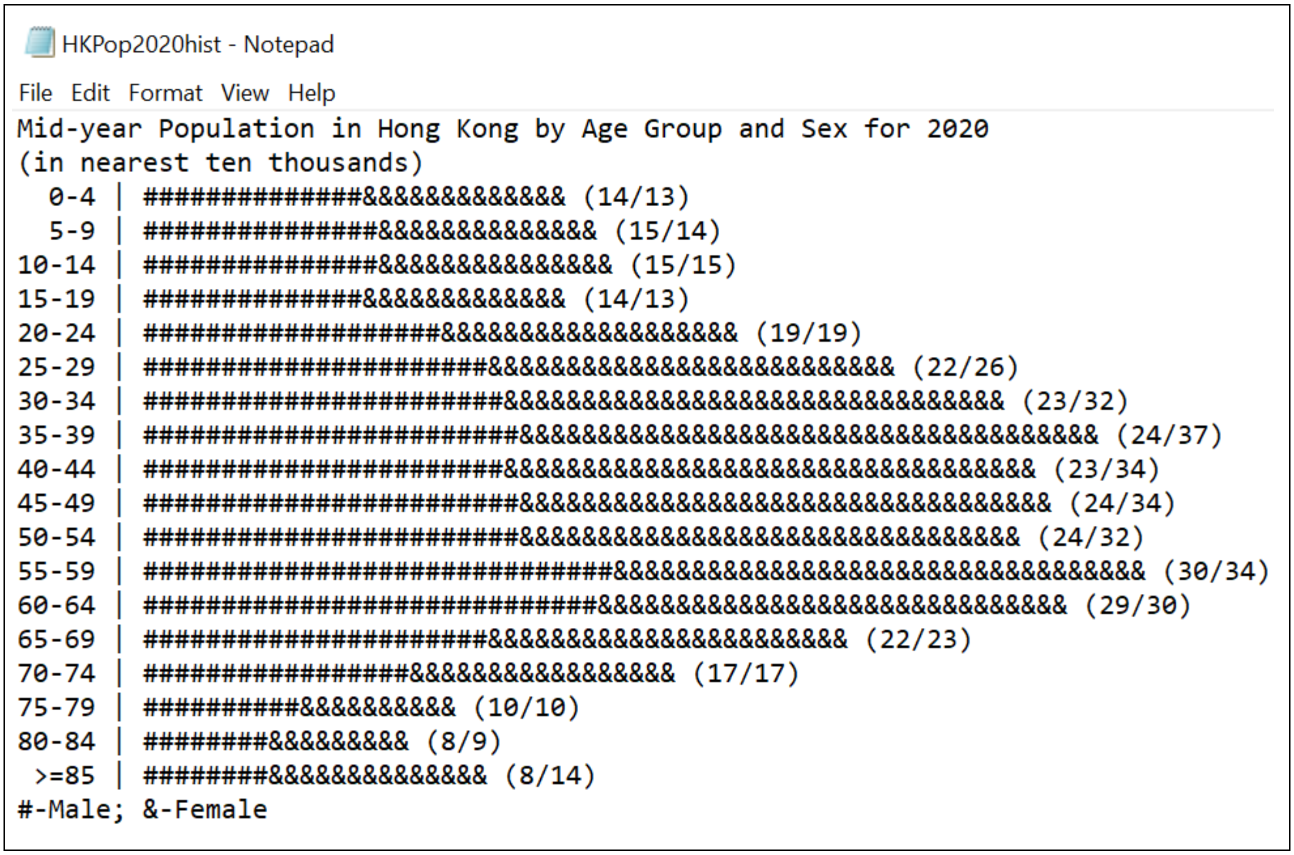
(The program is shown on the next page.)

PROGRAM:

```
# Exercise 3: Printing a Histogram to a Text File
# Written by F K Chow, HKU
# Latest Update: 2022/3/30

# Read the data of the 2020 mid-year population in HK from the file
with open('HKPop2020data.txt', 'r') as infile:
    line = infile.readline()    # Discard the title line
    line = infile.readline()    # Discard the header line
    data = []
    for line in infile:
        row = []
        for i in line.strip('\n').split(' '):
            row.append(i)
        data.append(row)

# Extract the statistics from the data and show them by printing a
# histogram to another file
with open('HKPop2020hist.txt', 'w') as outfile:
    print('Mid-year Population in Hong Kong by Age Group and Sex for '
          +'2020', file=outfile)
    print('(in nearest ten thousands)', file=outfile)
    for i in range(len(data)):
        num_m = int(round(int(data[i][1])/10000))
        num_f = int(round(int(data[i][2])/10000))
        print('{:>5s} |'.format(data[i][0]), end=' ', file=outfile)
        print('{:s}{:s}'.format('#'*num_m, '&'*num_f), end=' ',
              file=outfile)
        print('({:d}/{:d})'.format(num_m, num_f), file=outfile)
    print('#-Male; &-Female', file=outfile)
```

(The output is shown on the next page.)

OUTPUT:



```
Mid-year Population in Hong Kong by Age Group and Sex for 2020
(in nearest ten thousands)
  0-4 | ##############&&&&&&&&&&&&& (14/13)
  5-9 | ###############&&&&&&&&&&&&&& (15/14)
10-14 | ###############&&&&&&&&&&&&&&& (15/15)
15-19 | ##############&&&&&&&&&&&&& (14/13)
20-24 | ###################&&&&&&&&&&&&&&&&&&& (19/19)
25-29 | ######################&&&&&&&&&&&&&&&&&&&&&&&&&& (22/26)
30-34 | #######################&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&& (23/32)
35-39 | ########################&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&& (24/37)
40-44 | #######################&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&& (23/34)
45-49 | ########################&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&& (24/34)
50-54 | ########################&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&& (24/32)
55-59 | ##############################&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&& (30/34)
60-64 | #############################&&&&&&&&&&&&&&&&&&&&&&&&&&&&&& (29/30)
65-69 | ######################&&&&&&&&&&&&&&&&&&&&&&& (22/23)
70-74 | #################&&&&&&&&&&&&&&&&& (17/17)
75-79 | ##########&&&&&&&&&& (10/10)
80-84 | ########&&&&&&&&& (8/9)
 >=85 | ########&&&&&&&&&&&&&& (8/14)
#-Male; &-Female
```

Exercise 4: Evaluating a Test with Arrays

AIM:

A test consisting of 20 multiple-choice questions with 5 possible choices (A, B, C, D, and E) is conducted for a group of 5 students. Write a Python program to evaluate the answers of these students using the following algorithm:

(a) Read the string of the correct answers to the questions from the user and store the answers into an array of characters.

(b) Read the string of the answers of a student from the user and store the answers into an array of characters.

(c) Construct a Boolean array to indicate whether the answer of the student to each question is correct.

(d) Use the array in (c) to count the number of correct answers and then print the results.

(e) Repeat steps (b) to (d) for each student.

You can assume that all the inputs are in the required format. Here are the sample input and output of this program:

```
Enter the correct answers to the MC questions:
EEDAECAEEEBCADDBCEEB
Enter the answers of Student-1:
CEACBBDBDBCBCEEADABB
Number of correct answers: 2
Answers to the following questions are correct:
2 20
Enter the answers of Student-2:
EEDAECAEEEBCADDBCEEB
Number of correct answers: 20
Answers to the following questions are correct: ALL
⋮
```

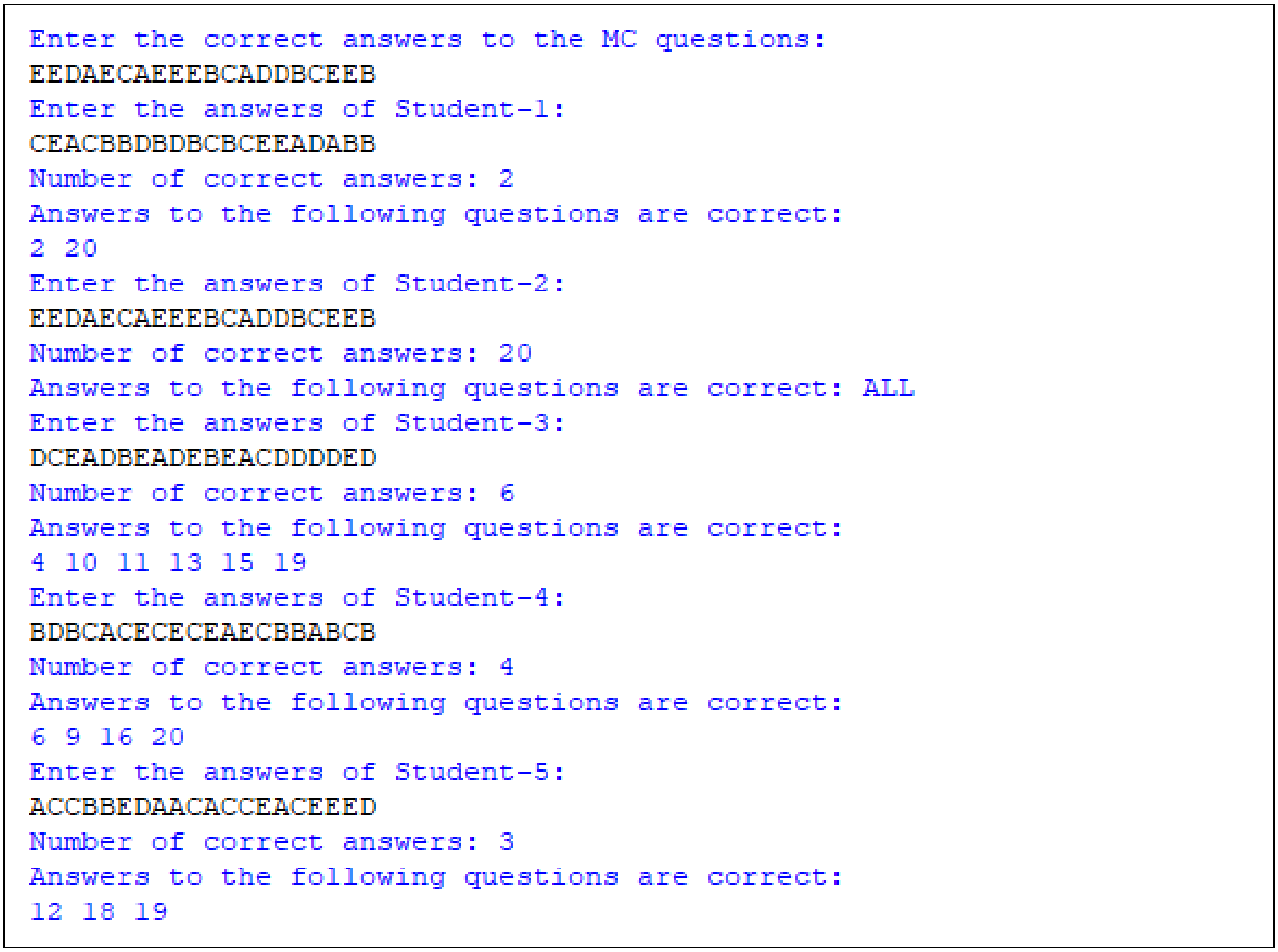
ALGORITHM:

```
1.  Start
2.  Import the numpy module
3.  Read the string of the correct answers to the questions from the user and store the answers into
    an array of characters
4.  Do the followings for each student:
    (a) Read the string of the student's answers from the user and store the answers into an array
        of characters
    (b) Construct a Boolean array to indicate whether the answer of the student to each question
        is correct
    (c) Use this array to count the number of correct answers and print the results
5.  End
```

PROGRAM:

```
# Exercise 4: Evaluating a Test with Arrays
# Written by F K Chow, HKU
# Latest Update: 2022/2/10

# Import the numpy module
import numpy as np

# Read the string of the correct answers to the questions from the
# user and store the answers into an array of characters
ans_str = input('Enter the correct answers to the MC questions:\n')
ans = np.array([ch for ch in ans_str])

# For each student, read the string of the student's answers from the
# user and store the answers into an array of characters, construct a
# Boolean array to indicate whether the answer of the student to each
# question is correct, use this array to count the number of correct
# answers and print the results
for i in range(1,6):
    stdans_str = input('\nEnter the answers of Student-'+str(i)+':\n')
    stdans = np.array([ch for ch in stdans_str])
    correct = (ans == stdans)
    ncans = np.sum(correct)
    print('Number of correct answers:', ncans)
    if ncans == ans.size:
        print('Answers to the following questions are correct: ALL')
    elif ncans > 0:
        print('Answers to the following questions are correct:')
        for j in range(ans.size):
            if correct[j]:
                print(j+1, end=' ')
        print()
```

OUTPUT:



```
Enter the correct answers to the MC questions:
EEDAECAEEEBCADDBCEEB
Enter the answers of Student-1:
CEACBBDBDBCBCEEADABB
Number of correct answers: 2
Answers to the following questions are correct:
2 20
Enter the answers of Student-2:
EEDAECAEEEBCADDBCEEB
Number of correct answers: 20
Answers to the following questions are correct: ALL
Enter the answers of Student-3:
DCEADBEADEBEACDDDDED
Number of correct answers: 6
Answers to the following questions are correct:
4 10 11 13 15 19
Enter the answers of Student-4:
BDBCACECECEAECBBABCB
Number of correct answers: 4
Answers to the following questions are correct:
6 9 16 20
Enter the answers of Student-5:
ACCBBEDAACACCEACEEED
Number of correct answers: 3
Answers to the following questions are correct:
12 18 19
```