Mid-Term- ECE 122 – Spring 2023

Closed book/notes- no calculator- no phone- no computer

NAME:
ID:

Problem	Score
1- General questions (32pts)	
2- Functions (22pts)	
3- String/List methods, I/O and Graphics (16pts)	
4- OOP (30pts)	
TOTAL (100pts)	

1- General questions [32 pts] For this question , the following variables have been defined:

name="pi"
pi=3.14159

[18pt] Complete the programs to obtain the corresponding outputs:

#[4.5] YOU MUST USE VARIABLES **name (or new_name) and pi**
#complete the single print instruction using embedded %s approach
print("2%s is %s"%(name,2*pi))
 	
new_name="2"+name
#complete the single print instruction using comma separated values
print(new_name , "is" , 2*pi)
#complete the single print instruction using one concatenation
print(new_name +" is "+ str(2*pi))
	2pi is 6.28318
	2pi is 6.28318
	2pi is 6.28318

#[2] complete (one line) to display the circle perimeter r=input("Enter circle radius: ") print("Perimeter is",2*pi*float(r))	**Enter circle radius: 2.5** **Perimeter is 15.70795**
#[2] complete the if condition r=float(input("Enter circle radius: ")) if pi*r**2>15: print("Good choice!\nyour circle area is strictly greater than 15")	**Enter circle radius: 2** ---- *new run* **Enter circle radius: 3** **Good choice!** **your circle area is strictly greater than 15**
#[2] complete the for instruction for i in range(-1,2): print("%s*%s=%s"%(i,name,i*pi))	**-1*pi=-3.14159** **0*pi=0.0** **1*pi=3.14159**

#[2] complete the for block
for c in str(pi):
if c!=".": #or if c==".": continue
print(c, end=" ") # print(c, end= " ")
	3 1 4 1 5 9

#[2] complete the for block pistr=str(pi) for i in range(len(pistr)): print(pistr[0:i+1])	**3** **3.** **3.1** **3.14** **3.141** **3.1415** **3.14159**

#[3.5] Using a while loop complete the program (**use** pi variable).
#Do **not** use a while True loop
count=1
result=float(input("Can you guess pi (5 digits)? "))
while result!=pi:
  result=float(input("Nope...try again= "))
  count=count+1
print("Well done after "+str(count)+" trials(s)")
	Can you guess pi (5 digits)? 3.14059
	Nope...try again= 3.14158
	Nope...try again= 3.14169
	Nope...try again= 3.14159
	Well done after 4 trial(s)

[2] For the question above, we now consider that the solution is obtained if the relative error between pi=3.14159 and the user input is less than 10^{-6}.
Give the new conditional statement for the while loop (one-line answer). Hint: |x-a|/|a| is the relative error between **x** (guess) and **a** (given)

while abs(result-pi)/pi>=1e-6:

[12pts] Provide the corresponding outputs to the following programs

approx_pi=str(pi) print(name,approx_pi) for i in [2,6,5]: approx_pi=approx_pi+str(i) print(name,approx_pi)	pi 3.14159 pi 3.141592 pi 3.1415926 pi 3.14159265

for i in range(4):
  for j in range(0,i+1):
  print(name,end=" ")
  print()
	pi
	pi pi
	pi pi pi
	pi pi pi pi

for i in range(-10,11):
  if (i%2==0 and i>0) or (i%2!=0 and i<0):
  print(i,end=" ")
	-9 -7 -5 -3 -1 2 4 6 8 10

for i in range(-10,11):
  if (i%2==0 or i>0) and (i%2!=0 or i<0):
  print(i,end=" ")
	-10 -8 -6 -4 -2 1 3 5 7 9

fruits=["banana","cherry","apple","kiwi","mango"] print(fruits[2:3]) print(fruits[0:4:2]) print(fruits[-1]) print(fruits[4:2:-1])	['apple'] ['banana','apple'] ['mango'] ['mango', 'kiwi']
n=50 i=5 s=0 while i<n: s=s+i i=i+10 print("i=",i) print("sum=",s)	i= 55 sum= 125

2- Functions [22pts]

[11pts] Complete this program to obtain the corresponding output

#[3] function
def my_pi(factor=1):
  return str(factor)+"pi is "+str(factor*3.14)
 	
#main program
print(my_pi())
print(my_pi(2))
	1pi is 3.14
	2pi is 6.28

#[3] function def power(p,n): print("%s**%s=%s"%(p,n,p**n)) return p**n #main program p,n=2,3 print("%s**%s=%s"%(p,n,power(p,n)))	**2**3=8** **2**3=8**

#[3] Given 2 ints, a and b, return their sum. However, sums in the #range 10..19 inclusive, are forbidden, so in that case just return 20.
# function 	
def sorta_sum(a,b):
  c=a+b
  if 10<=c<=19: c=20
  return c
#main program
print(sorta_sum(3, 4),sorta_sum(9, 4),sorta_sum(10, 11))
	7 20 21

#[2]Given an array of ints length 3, return the sum of all the #elements.
def sum3(nums):
  return sum(nums)
 	
#main program
print(sum3([1, 2, 3]), sum3([5, 11, 2]))
	6 18

[11pts] Find the corresponding outputs:

# [3] function 	
def func1(str, n):
  front_len = n
  if front_len > len(str):
  front_len = len(str)
  front = str[:front_len]
  result = ""
  for i in range(n):
  result = result + front
  return result
 	
# main program 	
print(func1('Chocolate', 2))
print(func1('Chocolate', 3))
print(func1('Ab', 3))
	ChCh
	ChoChoCho
	AbAbAb

# [2] functions 	
def fun1(c, d):
  return c-d
def fun2(a, b):
  return fun1(b, a)
 	
# main program 	
res = fun2(5, 10)
print(res)
	5

# [3] function 	
def func2(a, b, c):
  total=0
  ml=[a,b,c]
  for l in ml:
  if l==13: break
  total=total+l
  return total
# main program 	
print(func2(1,2,3))
print(func2(1,2,13))
print(func2(1,13,3))
	6
	3
	1

# [3] function 	
def h(base):
  result=1
  for i in range(base):
  result=result*(i+1)
  return result
# main program 	
for i in range(6):
  print("h(%s)=%s"%(i,h(i)))
	h(0)=1
	h(1)=1
	h(2)=2
	h(3)=6
	h(4)=24
	h(5)=120

3-String-list methods, I/O and graphics [16pts]

[4pts] Complete the program to get the corresponding outputs. Hint: methods that could be used are split, strip, lower

a,b,c=input("Three lucky letters: ").lower().split() print("Your lucky letters are",a,b,c)	**Three lucky letters: H A L** **Your lucky letters are h a l**
phrase="**ECE-122-MID-TERM**" print(phrase.strip("*").split("-"))	**['ECE','122','MID','TERM']**

[4pts] Find corresponding outputs

numbers1=[1,2,3,4] numbers2=numbers1.copy() numbers1.reverse() numbers2.append(5) print(numbers2+numbers1)	**[1, 2, 3, 4, 5, 4, 3, 2, 1]**

```
def square(a):
    return a**2
numbers1=[1,2,3,4]
numbers2=list(map(square,numbers1))
for i in range(4):
    print(numbers1[i],numbers2[i])
```

```
1 1
2 4
3 9
4 16
```

[4pts] Give the contents of the file2.txt after the program execution

```
f1=open("file1.txt","r")
f2=open("file2.txt","w")
lines=f1.readlines()
lines.reverse()
for line in lines:
    line.rstrip()
    temp=line.split()
    temp.reverse()
    temp2=""
    for w in temp:
        temp2=temp2+w+" "
    f2.write(temp2+"\n")
f2.close()
f1.close()
```

content of file1.txt

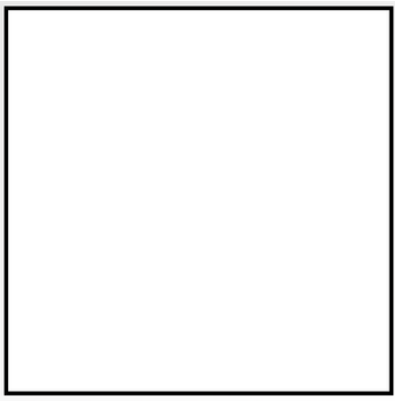
```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

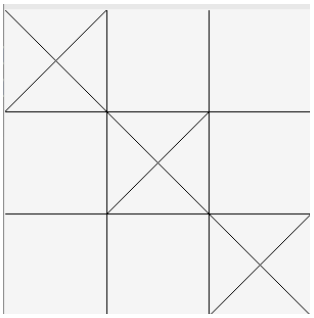
#content of file2.txt

```
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
```

[4pts] Drawing in the 300x300 grid canvas provided below.

```
from tkinter import *
root = Tk()
canvas = Canvas(root, width=300,height=300)
canvas.pack()
for i in range(3):
    canvas.create_rectangle(i*100,0,(i+1)*100,300)  #x1,y1,x2,y2
    canvas.create_rectangle(0,i*100,300,(i+1)*100)  #x1,y1,x2,y2
for i in range(3):
    canvas.create_line(i*100,i*100,(i+1)*100,(i+1)*100) #x1,y1,x2,y2
    canvas.create_line((i+1)*100,i*100,i*100,(i+1)*100) #x1,y1,x2,y2
root.mainloop()
```





4-OOP [30pts]

The goal of this section is to implement a class Exponential that can be used to represent a family of exponential functions of the form:

a*exp(b*(x-c))

where a,b,c are real numbers.

[6pts] Complete the __init__ and __str__ methods to get the corresponding output. Hint: there is only one instance variable which is the list **p**.

```
class Exponential:
  def __init__(self,a=1,b=1,c=0):
    self.p=[a,b,c]

  def __str__(self):
    return "[%s]*exp{[%s]*(x-[%s])}"%(self.p[0],self.p[1],self.p[2])


#main
e1=Exponential()
print(e1.p)
print(e1)
e2=Exponential(3,2,4)
print(e2.p)
print(e2)
```

```
[1,1,0]
[1]*exp{[1]*(x-[0])}
[3,2,4]
[3]*exp{[2]*(x-[4])}
```

[4pts] Complete the evaluate method

```
from math import exp
class Exponential:
  def __init__# no need to rewrite this method
  def __str__  # no need to rewrite this method
  def evaluate(self,x):  # to complete
    return self.p[0]*exp(self.p[1]*(x-self.p[2]))


#main (follow-up from previous main where e1 and e2 were defined)
print(e1.evaluate(1))
print(e2.evaluate(4))
```

```
2.718281828459045
3.0
```

[4pts] Complete the **scale** and **shift** methods. Hint: both methods work in-place.
The exponential scaled by a factor s becomes:

s*a*exp(b*(x-c))

The exponential shifted by a factor s becomes:

a*exp(b*(x-(c+s)))

```
 #continuation of class Exponential.....
 def scale(self,coef):  # to complete
   self.p[0]=self.p[0]*coef

 def shift(self,x0):     # to complete
   self.p[2]=self.p[2]+x0

#main (follow-up from previous main where e1 and e2 were defined)
e1.scale(3.0)
e1.shift(2.5)
print(e1.p)
print(e1)
e2.scale(3)
e2.shift(3)
print(e2.p)
print(e2)
```

```
[3.0,1,2.5]
[3.0]*exp{[1]*(x-[2.5])}
[9,2,7]
[9]*exp{[2]*(x-[7])}
```

[4pts] Write the method **derive** and **antiderive**. Both methods returns a new Exponential.
Hint: [a*exp(b*x)]'=a*b*exp(b*x) and F(a*exp(b*x))=(a/b)*exp(b*x)

```
 #continuation of class Exponential.....
 def derive(self):   # to complete
   a=self.p[0]*self.p[1]
   b=self.p[1]
   c=self.p[2]

   return Exponential(a,b,c)     #  given (do not change)

 def antiderive(self):   # to complete
   a=self.p[0]/self.p[1]
   b=self.p[1]
   c=self.p[2]

   return Exponential(a,b,c)     #  given (do not change)

#main
e1=Exponential(9,2,7)
print(e1.derive())
print(e1.derive().antiderive())
```

```
[18]*exp{[2]*(x-[7])}
[9.0]*exp{[2]*(x-[7])}
```

[6pts] Write the method **product** and **divide**. Both methods returns a new Exponential.
Hint: exp(a)*exp(b)=exp(a+b) and exp(a)/exp(b)=exp(a-b)

```
 #continuation of class Exponential.....
 def product(self,e0):   # to complete
   a=self.p[0]*e0.p[0]
   b=self.p[1]+e0.p[1]
   c=(self.p[1]*self.p[2]+e0.p[1]*e0.p[2])/b

   return Exponential(a,b,c)     #  given (do not change)
```

```
[27]*exp{[3]*(x-[5.5])}
[3.0]*exp{[1]*(x-[11.5])}
```

```
  def divide(self,e0):   # to complete
    a=1/e0.p[0]
    b=-e0.p[1]
    c=e0.p[2]
    return self.product(Exponential(a,b,c))    #  given (do not change)

#main
e1=Exponential(3,1,2.5)
e2=Exponential(9,2,7)
e3=e1.product(e2)
print(e3)
e4=e2.divide(e1)
print(e4)
```

[3pts] Write below the **integrate** method. Hint: Integration of f(x) between [a,b] is I=F(b)-F(a)
Requirement: you need to use the evaluate method

```
  #continuation of class Exponential.....
  def integrate(self,a,b):
    e0=self.antiderive()
    return e0.evaluate(b)-e0.evaluate(a)

#main
e1=Exponential(3,1,2.5)
print(e1.integrate(-5,-1))
```

```
0.088932897156512
```

[3pts] Write below the **multi_integration static** method. There are three input arguments: (i) a list of exponential, (ii) the lower bound a, (iii) the upper bound b
The method is going to perform the integration of each exponential in the list between [a,b] (must call the integrate method).

```
  #continuation of class Exponential.....
 @staticmethod
  def multi_integration(list_exp,a,b):
    v=[]
    for e in list_exp:
      v.append(e.integrate(a,b))
    return v


#main
family_exp=[]
for i in range(10):
  family_exp.append(Exponential(1,2,i*1.0))

values=Exponential.multi_integration(family_exp,-5,-3)
for i in range(len(family_exp)):
  print(family_exp[i],values[i])
```

```
[1]*exp{[2]*(x-[0.0])} 0.001216676123451937
[1]*exp{[2]*(x-[1.0])} 0.0001646592077745918 3
[1]*exp{[2]*(x-[2.0])} 2.2284200521690644e-05
[1]*exp{[2]*(x-[3.0])} 3.0158385893044754e-06
[1]*exp{[2]*(x-[4.0])} 4.081493696794276e-07
[1]*exp{[2]*(x-[5.0])} 5.523701054841028e-08
[1]*exp{[2]*(x-[6.0])} 7.475516467712868e-09
[1]*exp{[2]*(x-[7.0])} 1.0117011384978835e-09
[1]*exp{[2]*(x-[8.0])} 1.3691886012941457e-10
[1]*exp{[2]*(x-[9.0])} 1.8529952716048478e-11
```