

**San Francisco Bay University**

**CE450 Fundamentals of Embedded Engineering**

**2022 Fall Final Exam**

**Student Name: Soe Wunna ID: 19651**

**Part I Python Programming**

1. Write a function as a decorator of other function calls for the following operations

***def******trc1(g):***

*""" YOUR SOURCE CODE HRER """*

***@trc1***

***def sqr(x):***

*return x\*x*

***@trc1***

***def sum\_sqr(n):***

*""" YOUR SOURCE CODE HRER """*

*>>> sqr(3)*

***Calling*** *<function* ***sqr*** *at 0x7f73e7ce8620> on argument* ***3***

***9 # 9 = 3^2***

*>>> sum\_sqr(3)*

***Calling*** *<function* ***sum\_sqr*** *at 0x7f73e7c410d0> on argument* ***3***

***Calling*** *<function* ***sqr*** *at 0x7f73e7c41158> on argument* ***1***

***Calling*** *<function* ***sqr*** *at 0x7f73e7c41158> on argument* ***2***

***Calling*** *<function* ***sqr*** *at 0x7f73e7c41158> on argument* ***3***

***14 # 14 = 1^2 + 2^2 + 3^2***

*Hint: sqr(3) with a decorator @trc1 will be coming trc1(sqr)(3), likewise sum\_sqr(3) should be trc1(sum\_sqr)(3)*

def trc1(g):  
 def f(num):  
 return g  
 return f  
  
  
@trc1  
def sqr(x):  
 return x \* x  
  
  
@trc1  
def sum\_sqr(n):  
 total = 0  
 for i in range(n + 1):  
 total += i \* i  
 return total  
  
  
print(sqr(3))  
print(sum\_sqr(3))

1. Generate a function to implement the following operations

***def*** ***verify\_add(m, ls):***

*"""Returns True if addition of any two different elements in ls is m.*

*>>> verify\_add (100, [1, 2, 3, 4, 5])*

*False*

*>>> verify\_add (7, [1, 2, 3, 4, 2])*

*True # 7 = 3 +4*

*>>> verify\_add (10, [5, 5])*

*True*

*>>> verify\_add (151, range(0, 200000, 3))*

*False*

*>>>verify\_add(50004, range(0, 200000, 4))*

*True # 50004 = 50000 + 4*

*"""*

*""" YOUR SOURCE CODE HRER """*

def verify\_add(m, ls):  
 values = {}  
 for index, value in enumerate(ls):  
 if m - value in values:  
 return True  
 else:  
 values[value] = index  
 return False  
  
  
print(verify\_add(100, [1, 2, 3, 4, 5]))  
print(verify\_add(7, [1, 2, 3, 4, 2]))  
print(verify\_add(10, [5, 5]))  
print(verify\_add(151, range(0, 200000, 3)))  
print(verify\_add(50004, range(0, 200000, 4)))

1. Write a function to implement deep-reverse for taking a (possibly deep) tuple argument and reverses it including deep tuple element.

***def deep\_rvrs(tup):***

*"""Reverses tuple with possible tuple elements*

*>>> a = (11, 12, 13, 14)*

*>>> deep\_rvrs (a)*

*(14, 13, 12, 11)*

*>>>tpl = (11, (12, (13,113), 14), 15)*

*>>> deep\_rvrs (tpl)*

*(15, (14, (113, 13), 12), 11))*

*"""*

*""" YOUR SOURCE CODE HRER """*

def deep\_rvrs(tup):  
 return tuple(deep\_rvrs(x) if isinstance(x, tuple) else x  
 for x in reversed(tup))  
  
  
a = (11, 12, 13, 14)  
print(deep\_rvrs(a))  
  
tpl = (11, (12, (13, 113), 14), 15)  
print(deep\_rvrs(tpl))

1. Write a Fibonacci class to calculate next number in the ***'Fibonacci'***class by the

***'nxt'*** method. In this class, the ***'val'*** member is a ***'Fibonacci'*** number. The ***'nxt'*** method will return a ***'Fibonacci'*** object whose value is the next number in Fibonacci series.

***class*** *Fibonacci ():*

*"""A Fibonacci number.*

*>>> a = Fibonacci():*

*>>> a*

*0*

*>>> a.nxt()*

*1*

*>>> a.nxt().nxt()*

*1*

*>>> a.nxt().nxt().nxt()*

*2*

*>>> a.nxt().nxt().nxt().nxt()*

*3*

*>>> a.nxt().nxt().nxt().nxt().nxt()*

*5*

*>>> a.nxt().nxt().nxt().nxt().nxt().nxt()*

*8*

*"""*

***def*** ***\_\_init\_\_(self):***

*self.val = 0*

***def******nxt(self):***

*""" YOUR SOURCE CODE HRER """*

***def*** ***\_\_repr\_\_(self):***

*return str(self.val)*

class Fibonacci:  
 def \_\_init\_\_(self):  
 self.val = 0  
 self.next\_number = 1  
  
 def nxt(self):  
 ans = Fibonacci()  
 ans.val = self.next\_number  
 ans.next\_number = self.val + self.next\_number  
 return ans  
  
 def \_\_repr\_\_(self):  
 return str(self.val)  
  
  
a = Fibonacci()  
print(a)  
print(a.nxt())  
print(a.nxt().nxt())  
print(a.nxt().nxt().nxt())  
print(a.nxt().nxt().nxt().nxt())  
print(a.nxt().nxt().nxt().nxt().nxt())  
print(a.nxt().nxt().nxt().nxt().nxt().nxt())

*Hint: A new* ***'Fibonacci'*** *object is needed to create and assign****'val'*** *and****'pre'*** *members within****'nxt'*** *method.*

1. Create a class *'****Student'*** first and construct objects with student ***'name'*** and ***'number'*** of course(s) she/he is taking in the current semester. The following operations can be allowed by using magic methods (*'dunder'* method), such as ***\_\_add\_\_(), \_\_str\_\_(), \_\_repr\_\_(), \_\_lt\_\_(), \_\_eq\_\_(), \_\_ne\_\_(), and \_\_gt\_\_()*.**

***class*** *Student():*

*"""*

*>>> a= Student ('Peter', 3)*

*>>> b= Student ('Mike', 4)*

*>>> c= Student ('John', 5)*

*>>> d= Student ('Kelvin', 3)*

*>>> a+b+d*

*10*

*>>> a!=d*

*False*

*>>> b<c*

*True*

*"""*

class Student:  
 def \_\_init\_\_(self, name: str, number: int):  
 self.name = name  
 self.number = number  
  
 def \_\_add\_\_(self, other\_student):  
 return Student("", self.number + other\_student.number)  
  
 def \_\_lt\_\_(self, st):  
 return self.number < st.number  
  
 def \_\_eq\_\_(self, st):  
 return self.number == st.number  
  
 def \_\_str\_\_(self):  
 return str(self.number)  
  
 def \_\_repr\_\_(self):  
 return str(self)  
  
 def \_\_ne\_\_(self, st):  
 return self.number != st.number  
  
 def \_\_gt\_\_(self, st):  
 return self.name > st.number  
  
  
a = Student('Peter', 3)  
b = Student('Mike', 4)  
c = Student('John', 5)  
d = Student('Kelvin', 3)  
  
print(a + b + d)  
print(a != d)  
print(b < c)

**Part II Embedded System Design Theory**

1. Describe the application areas of the real time operating system (RTOS)

**Real-time operating system (RTOS)** is an operating system which is mainly aimed to serve real-time applications that process data as it comes in, without any buffer delay.

It is widely used in the following areas.

* Airlines reservation system
* Air traffic control system
* Systems that provide immediate updating
* Used in any system that provides up to date and minute information on stock prices.
* Defense application systems like RADAR.
* Networked multimedia systems
* Command control systems
* Internet telephony
* Anti-lock brake systems
* Heart pacemaker

1. Explain why the middleware is needed and where

Middleware is a software, providing common services and capabilities to applications outside of what’s offered by the operating system. The reason why it is needed is to bridge connections between different applications, platforms or services. The middleware plays a critical role in transforming and parsing data so each platform can communicate with one another.

Middleware is crucial in data management, cloud computing, application services, messaging, authentication, and API management because it helps software developers build applications more efficiently and keeps application development process quick and cost-effective. Middleware can support application environments that work smoothly and consistently across a highly distributed platform. Furthermore, it acts like the connective tissue between applications, data, and users.

1. Describe each component’s function of any operating system

Windows operating system is made up of various components. These are;

* Management of processes

The process management component of an operating system is generally used to monitor and handle various processes that are executing on the operating system at the same time.

* File Administration

File management in an operating system is responsible for the maintenance (or management) of files. It is also known as a type of computer software that is responsible for organizing and managing data files.

* Network Management

The procedure of administering as well as maintaining computer networks is known as network management. It comprises management of performance, failure analysis, network provisioning and quality of service assurance.

* Main Memory Management

A huge array of storage or bytes with an address is referred to as main memory. A series of reads or writes of specified memory addresses is utilized in the memory management process.

* Management of Secondary Storage

Secondary storage serves as a backup for the main memory. Assemblers, as well as compilers, are saved on a disc until they are fetched into memory and processed on the disc.

The characteristics of secondary-storage management are storage allocation, free space management and disc scheduling.

* Management of I/O Devices

In computer I/O, the operating system is responsible for managing and controlling I/O processes and devices. Because the functions and speeds of devices attached to the computer (I/O devices) vary so considerably, multiple ways for controlling them are required. These methods make up the operating system's I/O subsystem, which shields the rest of the system from the intricacies of controlling I/O devices.

* Security Management

Security management guarantees that the operating system authorizes the use of operating files, memory, CPU, as well as other hardware resources.

* Command Interpreter System

A command interpreter is defined as a component of a computer operating system that is responsible for the interpretation as well as the execution of interactively entered or program-generated commands. The command interpreter is known as the shell in various operating systems.

1. What general functions are there in any of device drivers, including the description for each?

* The device driver provides the rest of the operating system with the software interface to a given device or device class.
* The upper layer recognizes which commands are required to control a particular device or device class.
* The device driver builds I/O requests containing device commands, and sends them to the adapter device driver in the sequence needed to operate the device successfully. The device driver cannot manage adapter resources or give the command to the adapter. Specifics about the adapter and system hardware are left to the lower layer.
* The device driver also provides recovery and logging for errors related to the device that it controls.
* The operating system provides several kernel services that let the device driver communicate with adapter device driver entry points without having the actual name or address of those entry points.

*Link:* ***https://github.com/SoeWunna29/Final-Exam.git***