HW03:

SORAY CENGİZ
161024077

12.1.1 truncated-exp-rv-samples-Monte-Carlo-octave-python-c-cpp

12.1.1.2 prob_waits_for_more_than_10_min_test.py

In Python:

```
18
19  import numpy  as np
20  import math   as ml
21  import random as rnd
22  """
23    Q:
24      What is an import alias?
25      What would we do to import just a single submodule
26        inside a module with an alias?
27      Is it possible to import multiple modules with a single
28        import command?
29  """
```

24.satır cevabı:

Import alias bir modülün kullanımı için yapılan bir isim değişikliğidir, genellikle uzun olan modül isimlerine kısa bir isimle kullanmak için kullanılır; Örneğin 19.satırda bir import alias gözükmektedir. Kod içerisinde numpy modülünün bize kazandıracakları fonksiyonları kullanabilmek için numpy.sin(x) yazmak yerine np.sin(x) yazabiliriz.

“from numpy import sin as s” şeklinde de bir modüle function alias yapılabilmektedir. Fakat sin kullanmak yerine s kullanmak çok anlamlı gözükmemektedir. Kod içerisinde numpy.sin(x) yerine s(x) kullanmamız mümkün olacaktır. Bu kullanım tüm numpy modülünü import eder mi? Cevap hayır sadece ilgili modülün ilgili fonksiyonunu yani np.sin’i import eder.

25.satır cevabı:

Yazacağımız bir modül içerisine daha önce yazılmış olan bir submodül’ü import etmek “import sub_module as s_m” şeklinde yapılabilir, ancak kullanılan submodüle ile aynı directory içerisinde “__init__.py” dosyası yer almak zorundadır.

26.satır cevabı:

```
1   import module as m
2   import numpy, getopt, sys, math
3
4   from numpy import cos
5   from math import cos as cm
6
7   from numpy import arange, linspace, array
8   from os import system
9
10  print('sin -> with numpy : ', numpy.sin(1))
11  print('sin -> with math  : ', math.sin(1))
12
13  print('')
14
15  print('cos -> with numpy : ', cos(1))
16  print('cos -> with math  : ', cm(1))
17
18  print('')
19  system('echo Inside Directory : ')
20  system('ls')
21
```

Cevap Evet mümkün, yukarıda küçük bir örnek ile bu durum gözlemlenmiştir. 2.satırda yazılan çoklu import modülleri 10. Ve 11.satırda sorun çıkarmamıştır. Bunun haricinde modül içindek fonksiyon veya class’ları driver koduna çağırmanın çeşitli yöntemleri gösterilmiş ve denenmiştir.

Output:

```
soray@soray-VirtualBox:~/Desktop/deneme/q1$ python3 driver.py
sin -> with numpy :  0.8414709848078965
sin -> with math  :  0.8414709848078965

cos -> with numpy :  0.5403023058681398
cos -> with math  :  0.5403023058681398

Inside Directory :
driver.py  module.py  modules  __pycache__
```

```
30  rnd.seed()
31  """
32    Q:
33      What does the command above do? If I did not have this
34        one in the octave implementation, how must I have decided
35        that a seed determination is not necessary to be carried
36        out by the developer?
```

32.satır cevabı:

Seed yöntemi rastgele sayı üretecini başlatmak için kullanılır. Rastgele sayı üretecinin, rastgele bir sayı üretebilmesi için, bir seed'e ihtiyacı vardır. Pyton'da 30.satır yazılmamış olsaydı da kod çalışacaktı çünkü default olarak python bu seed için sistem saatini kullanmaktadır.

rnd.seed() tam olarak bir seed belirliyor bu seed'in ne olduğunu biz bilmiyoruz bu satır kodu her çalıştırdığımızda farklı bir tohum üretecektir eğer "rnd.seed(1)" yazsaydım kodu her çalıştırdığımda aynı sonucu elde edecektim çünkü tohum sabit hep belirlenen bir sayıyı bize veriyor (0 ile 1 arasında oluşturuyor). "rnd.random"'un üreteceği sayı bu seed'e bağlıdır.

Bu kod her çalıştığında farklı bir sonuç verecektir, örneğin 3.çalıştırmada elde edilen sonucu tekrar elde etmek istiyor olalım, bunun için bize hangi tohum ile bu random sayıların üretildiği bilgisi gerekecektir eğer bu bilgiyi bilirsek ve 30.satırı rnd.seed(x) şeklinde düzenlersek x burada 3.çalıştırmada kullanılan seed (tam olarak seed değil) değeridir. Aynı sonuca tekrar ulaşmak istediğimiz durumlarda bu yöntem kullanılacaktır. Aynı random sayıları tekrar tekrar üretmek için de seed kullanılıyor.

Sürekli aynı sonuçları gözlemlemek istiyorsam seed determination önemli değildir. Fakat python default olarak gidip seed değiştiriyor, seed'i bizim sabitlememiz lazım bu da aşağıda basit bir örnek ile gösterilmitir. Aslında tam bir random sayı üretmiyoruz (Sözde random sayı üretimi söz konusu oluyor)

```
24  rnd.seed()
25  rnd.seed(1)
26  print(rnd.random())
27
28  rnd.seed()
29  rnd.seed(1)
30  print(rnd.random())
```

Yukarıdaki kod parçasında 24. Ve 28.satırlar comment in yapılsın üretilecek olan 2 sayı birbirinin aynısı olacaktır ve kod her çalıştırıldığında aynı 2 sayı ekrana bastırılacaktır, 24. 27. 28 comment in yapılırsa her iki random değer farklı olacaktır fakat kod her çalıştırıldığında bu değerler aynı kalacaktır, bunlar gibi birkaç olasılık daha denenmiştir sonuçlar gözlemlenmiştir.

```
37    What is the default seed for the random number generator
38      when called as above? Why would we refrain from feeding
39      a seed with a constant value to the generator? Also
40      what does the above call tell us about the seed value
41      determined being global or local? What may be the
42      advantages or disadvantages associated with either
43      alternative (i.e., global or local seed value)?
```

37.satır cevabı:

"rnd.seed()" içinde bir şey yazılmamış ise default olarak sistem saati kullanılır. Rastgelelik kaynakları işletim sistemi tarafından sağlanıyor ise, bunlar sistem saati yerine kullanılır. (os.urandom())

“np.seed(x)” neden x kullanmayız, bir constant ile belirleme yapmıyoruz çünkü yapsaydık o seed’e bağlı kalarak aynı random sayıları elde edecektik, istenilen durum bunun tersine kod her çalıştığında farklı random sayı üretmemiz ve bununla beraber kod içerisinde kaç tane random sayı üretilecek ise bu sayılar da birbirlerinden farklı olsun istenmektedir.

30.satırda belirlenmiş seed değeri globaldir aşağıda basit bir örnek ile gözlemlenmiştir.

```
24  rnd.seed(1)
25
26  def fnc():
27    rnd.seed(2)
28    print(rnd.random())
29
30  print(rnd.random())
31  print(rnd.random())
32
33  fnc()
34
```

Yukarıdaki 27.satırı yorum satırına alırsak yaparak 33.satırda üretilecek olan random sayı bir önceki sonuca göre değişiklik gösterecektir ve her çalışmada 3 sayı da bir önceki çalıştırmada elde edilen sayılar ile aynı olur 24.satır 28.satırdaki random sayıya da seed sağlamıştır.

24.satırı yorum satırına alıp 27.satırı yorum satırından çıkartırsak sadece 28.satırda oluşturulan random sayı sürekli aynı kalır diğerleri sürekli birbirinden farklı olacaktır. Yani fonksiyon içerisinde local olarak oluşturulan seed global gibi davranamaz 30 ve 31.satırlara seed sağlayamaz, 30 ve 31 default seedler üzerinden işletilir (saat üzerinden).

Fonksiyon içerisinde yapılacak olan işlemler yani burada fonksiyon içerisinde bir random sayı üretmek isteyelim ve bu üretilecek sayı tüm derlemelerde hep aynı kalsın o üretilen sayıyı kullanalım her çalışmada demek istiyorsak local seed işimize yarar fonksiyon dışındaki random sayı üretimine karışmaz. Bunu bazı uygulamalarda isteyebiliriz.

Global ise her yere seed sağlayabilir fonksiyon, class, lamda içlerine dahi tek bir seed belirlenerek tüm sayılar bu seed bağlamında üretilebilir, tabi rnd.random(x) içine x yazıp yazmamak sizin isteğinize bağlıdır sürekli aynı sayıların (her bir derlemede) üretilmesini istersek içine bir şey yazmak lazım.

```
46  """
47    Q:
48      What do we do for line continuation for a single command?
49        We have examples below for assignments.
50  """
51  no_mc = 100000
52  lam   = 1. / 10
53  time_to_wait = \
54    10.
55  total_min_in_hour = \
56    60.
57
```

47.satır cevabı:

Python’da tek satıra sığdıramadığımız işleme aşağıda devam etmek istiyorsak ”\” yazmamız gerekmektedir indentation gerekli mi? Cevap Hayır ama kodun okunması için mutlaka konması gerekmektedir 54.satır indentation olmadan yazılsa çalışır mıydı? EVET. “…” matlab için bu syntax kullanılmalıdır. C ve C++’da sorun yok direkt aşağıya geçebiliriz indentation koymak lazım anlamak için. Perl’de “\” olsa da olmasa da çalıştı.

```
59    Q:
60      What is the list data structure in python?
61      What is a generator (I am not referring to the random
62        number generator in here) in python?
63      How will the random variable samples generated with
64        "rnd.random()" below be distributed? What will be
65        probabilistic distribution associated?
66      And what happens when we multiply the samples generated
67        by "rnd.random()" with a constant, as in the python
68        generator below? How will the resulting samples be
69        distributed now?
70  """
```

60.satır cevabı:

Liste basitçe birçok değişkeni bu değişkenler farklı tiplerde olabilirler (int, char, list, tuple vs.) bünyesinde tutabilen veri tipidir.

61.satır cevabı:

Generator syntax Python'da tek satırda birçok şeyi halletmemize olanak sağlar; iteratif olarak bir şeyleri liste içerisine sıkıştırır. (Yield da bunu yapıyordu)

Aşağıda basit bir kod parçası generator syntax kullanımının mantığını tekrardan hatırlamak için yazılmıştır;

```
37  import numpy as np
38
39  print(' How can we use Generator Syntax in PYTHON \n')
40
41  my_list = [item  for item in range(10)]
42  print('my_list       : ', my_list)
43  print('type(m_list)  :' , type(my_list))
44
45  my_list_A = [[n, l] for n in range(2) for l in range(2)]
46  print('my_list_A     : ', my_list_A)
47  print('')
48
49  print(' BASIC EXAMPLE ')
50
51  m_array = np.array([])
52
53  my_list_B = [np.append(m_array, item)  for item in range(3)]
54
55  print('my_list_B     : ', my_list_B[:])
56
57  print('my_list_B[0]  : ', my_list_B[0])
58  print('my_list_B[1]  : ', my_list_B[1])
59  print('my_list_B[2]  : ', my_list_B[2])
60
61  print('my_array      : ', m_array)
62
63  print('')
64
65  def use_yield():
66    for item in range(10):
67      yield item
68
69  print(' USING YIELD KEYWORD ')
70
71  a = use_yield()
72  print('next : ', next(a))
73  print('next : ', next(a))
74  print('next : ', next(a))
75  print('next : ', next(a))
76  print('next : ', next(a))
```

Yukarıdaki kod parçasında 41. Ve 45.satırlarda kullanılan syntax aslında for sayesinde adım adım listeye append işlemine karşılık gelmektedir. Bunu tek satırda yapabiliyoruz.

Output:

```
 How can we use Generator Syntax in PYTHON

my_list       :  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
type(m_list)  : <class 'list'>
my_list_A     :  [[0, 0], [0, 1], [1, 0], [1, 1]]

 BASIC EXAMPLE
my_list_B     :  [array([0.]), array([1.]), array([2.])]
my_list_B[0]  :  [0.]
my_list_B[1]  :  [1.]
my_list_B[2]  :  [2.]
my_array      :  []

 USING YIELD KEYWORD
next :  0
next :  1
next :  2
next :  3
next :  4
```

63.satır cevabı:

"rnd.random()" kullanılarak 0 ile 1 arasında bir random sayı üretilecektir. Bu sayının her compile aşamasında farklı olacağı seed'e bağlıdır ve farklı olacaktır. Ayşe'nin tam olarak saat kaçta kafeye gelineceği bilinmiyor, saat 2 ile 3 arasında kalan bir saat diliminde kafeye kesin geleceğini biliyoruz, aynı zamanda Ali'yi kesin olarak bekleyeceğine de eminiz, 10 dakikadan fazla beklemeyecektir ve saat 3'den sonra da kafede bulunamayacaktır.

Ayşe bu 60 dakikalık aralıktan herhangi bir dakikada geleceği kesin ama hangi dakika olduğu belirsiz 60'ı random 0 ile 1 arasında bir sayı ile çarparsak, random bir dakika elde ederiz.

66.satır cevabı:

Buradaki örnekler no_mc tanedir ve hepsi sırayla bir listeye kaydedilmiştir, örneklerin her biri Ayşe'nin olası kafeye varış saatinin dakikasını söylemektedir (eğer 0 ilse Ayşe saat 2'de kafeye gelmiştir, eğer 60 ise Ayşe saat 3'de kafeye gelmiştir), ve bu örnekler [0, 60] arasındadır.

```
71 arrival_times = \
72   [ total_min_in_hour*rnd.random() for kk in range(no_mc) ]
73 """
74   Q:
75     What is the return type of "np.zeros()", as used below?
76       Is it one that corresponds directly to the matrix data
77       structure of octave? Or is it something else?
78     What are the dimensions for "result" below going to be?
79       How can we check the dimensions through the "shape"
80       attribute? And what is the type for the "shape" attribute
81       of "result"?
82 """
```

75.satır cevabı:

"np.zeros(no_mc)"'nin return type'ı ndarray'dir içerisindeki her bir değişken ise default float64 tipindedir.

76.satır cevabı:

Matrix tipinde değildir, "np.zeros((3,1), np.uint8)" bize 3x1 boyutunda sıfırlardan oluşan bir ndarray return edecektir. Her bir satır kendi başına da bir ndarray'dir, toplam 3 satır olduğundan ndarray of ndarray gibi bir yapı vardır ve içerisindeki "0" item'larının tipi de bizim belirlediğimiz uint8 tipinde olacaktır.

78.satır cevabı:

Python'da indexler 0'dan başlar (C,C++,Perl'de de böyle Matlab'da 1'den başlıyor). Result için indexler şöyle olacaktır; 0'dan no_mc-1'e kadar. [0,no_mc) gibi düşünebiliriz, no_mc dahil değil.

79.satır cevabı:

print(result.shape) ve print(len(result)) aynı sonucu yani no_mc' değerini verecektir. Print(type(result.shape)) sonucu bir tuple'dır. Tuple içerisinde listenin uzunluğunu bize return eder. Print(type(len(result))) sonucu ise direkt olarak int değer return eder.

```
83 result = \
84   np.zeros(no_mc)
85
86 """
87   Q:
88     What is the return type for each of "range()" and "len()"?
89     Could not I have utilized "np.arange()" instead of "range()"?
90       Are the return types for the stated functions the same?
91       If not how can they be used interchangably in the statement
92       below?
93     What does "range()" return with a single input argument?
94       Are there other possibilities for the input argument set of
95       "range()"?
96     Could I iterate through "arrival_times" in a different way,
97       possibly making use of the "in" keyword? Why do I not
98       prefer the indicated alternative in here?
99 """
```

88.satır cevabı:

Print(type(range(10))) bize bir range tipinde bir şey return edecektir, len ise bize direkt olarak listenin uzunluğunu integer olarak return eder.

89.satır cevabı:

Cevap Evet np.arange(0,10,1) ile range(10) for için aynı şeyi yapmaktadırlar. Fakat return ettikleri şeyler farklıdır.

100.satır for kk np.arange(0, len(arrival_times)) şeklinde de kullanılabilirdir. Arange for içerisinde kk'e np.uint64 tipinde değerler atıyor range ise int tipinde değerler veriyor (default olarak ikisi de int tipinde).

93.satır cevabı:

"range(10)" -> [0, 1, ….. 9]'a kadar bir range return eder içerisindeki her bir eleman int tipindedir.

"range(0,10,2)" -> [0, 2, …. 8]'şeklinde bir kullanım da mevcuttur, range'deki elemanlar ikişer ikişer artacaktır.

96.satır cevabı:

Bu soruda range based for yapmak zorundayız for each loop yapamayız (In kullanarak) sebebi tam olarak 134.satırdır.

"for kk, ll in zip(arrival_times, result):" (NOT: burada arrival_times ve result aynı uzunlukta olmalı) yazsaydık ve arrival_times[kk] gördüğümüz yere kk aynı amanda result[kk] gördüğümüz yere de ll yazarsak bir ihtimal çalışmasını beklerdik, fakat çalışmayacaktır

134.satır result[index] olmak zorundadır ll = 1 gibi bir kullanım olamıyor listenin elemanları for içinde değişemiyor yani bu kullanım ile değişemiyor.

İlle de in kullanarak for each loop'u kullanarak yapmak isteseydik 134.satıra bir çözüm bulabilirdik ancak ekstra kopyalama gerekecekti.

```
100 for kk in range(len(arrival_times)):
101   if arrival_times[kk] > total_min_in_hour - time_to_wait:
102     continue
103   else:
104     """
105       Q:
106         What is the common type for the "True" and "False" values
107           in python?
108     """
```

105.satır cevabı:

Her ikisi için de type(True) ve type(False) yazıp incelersek tip olarak "bool" return edecektir.

```
109     flag_valid_sample = False
110     """
111       Q:
112         Does it matter what "sample" is assigned to at this
113           point? What is special about the "None" type? Are
114           there any contexts where the usage of "None" perhaps
115           makes more sense?
116     """
```

111.satır cevabı:

117.satır yani "sample=None" olmasaydı algoritma sorunsuz çalışacaktı, None keyword'ü aslında boş bir değişkeni veya nesneyi tanımlamak için kullanılır. Python'da None keyword'ü bir nesnedir ve NoneType sınıfının bir veri türüdür. NOT: None olarak atanan tüm değişkenler aynı nesneyi işaret ederler. None bir 0, empty veya False ifadelerinden herhangi biri değildir. NoneType tipi sample değişkenine bazı özellikler kazandırıyor olabilir.

```
117     sample = None
118     while not flag_valid_sample:
119       u = rnd.random()
120       """
121         Q:
122           What is the difference between "1" and "1." in python?
123             Would the value of "x" below differ if I had used
124             "1" instead of "1." in the indicated assignment?
125           How is the call for the "log()" function different
126             compared to the call in octave? Are there any more
127             functions encapulated inside the "math" module?
128       """
129       x = - 1. / lam * ml.log(1. - u)
130       if arrival_times[kk] + x < total_min_in_hour:
131         flag_valid_sample = True
132         sample = x
133     if sample > time_to_wait:
134       result[kk] = 1.
135
```

121.satır cevabı:

"1" bir integer "1." ise bir floating number'dır. Burada 129.satıra bakarsak float / something' sonucu kesin olarak bir floating point number verecektir. Sağ taraftakinin ne tipte olduğunu bilmediğimizi düşünelim ve 1 / something yapalım sonucunda bize something'e bağlı olarak int veya float değer verecektir, something int ise bize integer değer verilecek örneğin something = 2 ise yanlış sonuç elde edeceğiz 0 elde edeceğiz. Something float ise 1 / 2. İse doğru sonucu 0.5'i elde edeceğiz. Bu bakımdan "1." tercih edilmiştir.

129.satır özelinde düşünürsek "1" kullansaydık ve sağ tarafın float olduğu bilindiği için int / float bize doğru sonucu yani floating point number verecektir.

125.satır cevabı:

Python'da math modülü içerisinde birçok log kullanımı için fonksiyonlar bulunmaktadır. (math.log, math.log2, math.log10, math.log1p) gibi kullanımlar mevcuttur bunların eşdeğerleri Matlab'da da vardır.

```
136 print("Probability of Waiting for more than 10 min:")
137 """
138   Q:
139     Notice that I did not have to code a function to sum up
140       the entries in "result", how is that possible? What is
141       "np.sum()" able to do in order to compute the stated
142       sum?
143     If you have a type that you defined yourself in python,
144       perhaps through a class definition of yours, how would
145       you enable the "len()" function to operate on it? Checking
146       the member functions for the type that "result" belongs to
147       would help in this case.
148 """
149 print ("  %.6f" % ( np.sum(result) / len(result) ))
```

138.satır cevabı:

"np.sum(result)" kullanarak result array'inin içindeki her şey toplanır ve toplam return edilir, 2 boyutlu bir array'in tüm elemanlarının toplamını yine aynı fonksiyon ile basit bir şekilde yapabiliriz eğer sadece satırdaki elemanları toplamak istersek np.sum(xx, axis = 0) yazarsak bir array return edecek ve bu return ediler array satır ve sütunlardaki itemlerin toplamlarını içerecek return edilen bu array içerisindeki 0.elemanı çekip alırsak sadece satırlardaki elemanların toplamına erişiriz.

143.satır cevabı:

Bir __len__() fonksiyonu yazılır (overload edilir), return len(self.my_list) diyerek class'ın member variable olarak tuttuğu listenin uzunluğunu driver içerisinde yine instance.len() diyerek yazdırabiliriz.

Output:

```
soray@soray-VirtualBox:~/Desktop/433_hw/hw_3$ python3 prob_waits_for_more_tha
n_10_min_test.py
Probability of Waiting for more than 10 min:
  0.259370
```

12.1.1.4 prob_waits_for_more_than_10_min_test.cc

In C++:

```
19  #include <cstdio>
20  #include <cassert>
21  #include <chrono>
22  #include <random>
23  #include <cmath>
24  #include <vector>
25
26  #define NO_MC  100000
27
28  /*
29   * Q:
30   *   What is the advantage of having the opportunity
31   *     to make use of namespaces or even nested namespaces?
32   * */
33  namespace os_test {
34    namespace elm_218_probability {
```

29.satır cevabı:

Namespace'ler genel olarak Encapsulation prensibine uymak için kullanılır bu prensip basitçe kod içerisinde kullanılan fonksiyonlar, classlar veya değişkenler mümkün olduğunca main tarafından erişilmesi zor olsun fonksiyonlar açık seçik durmasınlar neden? void output() fonksiyonu bir çok sebepten dolayı yazılabilir kullanılabilir, kod içerisinde birden çok test yazacağımızı farz edelim ve bu testlerin bazıları için output fonksiyonu yazılması gereksin dolayısı ile her bir test ayrı bir namespace altında yapılsın ki bu testlerin kullanacağı fonksiyonlar da bu namespace altında yazılsın karmaşa yaşanmasın, fonksiyon overloading'i kullanarak da halledebilirdik ama (%100 bir çözüm sunamazdı). Bu gibi birçok sebepten namespace'ler hatta nested namespace'ler kullanılmaktadır. Nested namespace namespace altında o namespace içinde tanımlanan alt namespace'dir bazı fonksiyonlar veya typedefler veya input parametreleri için nested namespace'lere başvurulabilir.

```
35
36    class UniformlyDistributedRVGenerator
37    {
38    /*
39     * Q:
40     *   Do we need this "private:" indicator? Can we do without it?
41     * */
42    private:
43      unsigned
44        seed_; // seed for the random number generator
45      std::vector< std::default_random_engine >
46        v_generator_;    // vector of generators
47
```

39.satır cevabı:

Kodu nasıl yazmak istediğimize bağlı olarak ihtiyacımız var veya yok, bu olmadan da olur muydu? Olurdu. Private'ı kullanma amacımız private altında tanımladığımız değişkenleri veya fonksiyonları dış dünyadan saklamak istememizdir. Neden saklamak istiyoruz? Objelerin tuttuğu variable'ler değiştirilmemelidir direkt olarak obj.number = 10 yapılmamalıdır (mümkün olduğunca) bu variable'lar private ise zaten bunu yapmak mümkün olmayacaktır ancak getter, setter yazarsak bunlara ulaşabiliriz veya değiştirebiliriz.

```
47
48    // distributions
49    std::vector< std::uniform_real_distribution<double> >
50      v_dist_uniform_; // vector of indicated distributions
51
52    /*
53     * Q:
54     *   What are the advantages and disadvantages to keeping
55     *     the generator and the distribution in vectors as
56     *     indicated above? As a hint, one advantage will
57     *     become clear in the constructor.
58     * */
59
```

53.satır cevabı:

std::uniform_real_distribution<double> [A,b) aralığına eşit olarak dağıtılmış, yani olasılık yoğunluk fonksiyonuna göre dağıtılan rastgele kayan kayan nokta değerlerini üretir.

```
60  public:
61    UniformlyDistributedRVGenerator
62    ()
63    {
64      /*
65       * Q:
66       *   What is the "epoch" in this context?
67       *   Is the seed thus set through the current time
68       *     information a local or global one?
69       *   What would be advantages to making use of a local
70       *     seed value?
71       *   Should the seed for a random number generator be set
72       *     every time a sample obeying a certain distribution
73       *     is created? Or is this a gruesome mistake? Is this
74       *     how we do it in the current source code?
75       * */
```

65.satır cevabı:

Epoch genel olarak bir saatin başlangıç noktasını temsil eder. C++11'da chrono'da birden fazla saat vardır, bu algoritma da kullanım amacı Python'da seed default olarak o anki saat'e bakarak bir seed belirlemesi mümkün oluyordu. Python'daki bu yapıya benzer bir yapı C++'da chorono içerisindeki anlık saat üzerinden bir değişken kullanarak bir seed belirlemesi yapmak olacaktır.

67.satır cevabı:

Bu sorunun cevabı local olacaktır seed zaten constructor içerisinde her bir instance için belirlenmek durumunda olacaktır.

71.satır cevabı:

Her bir instance için bir seed belirlemesi mümkün olacaktır ve her bir instance için seed'ler farklı olacaktır. Main kısmında veya her hangi bir fonksiyonda bu class'ın instance'si oluşturulacaktır 273.satırda 1 kere instance oluşturulmuş ve bu instance üzerinden algoritma işletilmiştir, seed her bir sample'da tekrar belirlenmemelidir.

Seed, generator tarafından oluşturulan önceki değer numarasıdır, ilk seed için her hangi bir önceki değer olamayacağından mevcut saat sistemi kullanılacaktır.

```
76      seed_
77        = std::chrono::high_resolution_clock::now()
78            .time_since_epoch().count();
79
80      /*
81       * Q:
82       *   What are the arguments to the "emplace_back(...)"
83       *     member function of a vector? Do the number and
84       *     types of the indicated input arguments differ
85       *     for different template parameters for std::vector?
86       *   "emplace_back(...)" makes use of a paradigm called
87       *     perfect forwarding. What is this concept?
88       *   Do the "emplace_back(...)" calls below involve
89       *     object copying operations? If yes, is there an
90       *     extra call we may utilize to prevent copy operations
91       *     during intended in-place constructions in a vector?
92       * */
93      v_generator_.clear();
94      v_generator_.emplace_back( seed_ );
95
96      // initialize commonly used distributions
97      v_dist_uniform_.emplace_back( 0.0 , 1.0 );
98    }
99
```

81.satır cevabı:

emplace_back ve push_back vector için aynı işi yapmaktadırlar, her ikisi de ilgili vector'un en son kısmına bir item eklerler. Vector bir class'dır dolayısı ile bu iki member fonksiyonun birbirinden farkı şudur;

push_back sırası ile ne yapmaktadır?

Constructor çağırılacak ve geçici bir obje oluşturulacaktır, bu obje vector içine yapıştırılacak kopyalama işlemi bittikten sonra bu geçici obje öldürülecek (destructor).

emplace_back ise geçici bir obje oluşturmuyor direkt olarak vector içerisinde constructor oluşturuyor veya constructor argümanı ilgili vektör için yerinde oluşturuluyor. Geçici objeye gerek kalmıyor constructing ve destructing işleri ile uğraşılmıyor. Perfect forwarding işliyor.

Perfect Forwarding, aşırı kopyalamayı azaltır ve Lvalue ve Rvalue değerlerini ayrı ayrı ele almak için aşırı yük yazma ihtiyacını azaltarak kodu basitleştirir.

Mesela std::make_shared<Container>(5) obj --> burada make_shared perfect forwarding'i kullanıyor. Yaptığı şeyler bir container instance oluşturmak (allocation), ardından ona bir ilk değer kazandırmak (constructor) tek satırda ikisi de yapılıyor.

```
100   /*
101    * Q:
102    *   What does the following syntax mean for the destructor?
103    *     Are there any other reserved keywords that could be
104    *     used instead of "default"? Why is "default" the one
105    *     that makes sense in here?
106    * */
107   ~UniformlyDistributedRVGenerator()
108     = default;
```

101.satır cevabı:

Destructor {} süslü parantez içerisinde birisi tarafından da yazılabilir ama yazmadan da bizim için derleyici defult bir destructor oluşturuyor. Default keyword'ü sayesinde yıkım işini compiler'a bırakıyoruz.

Class instance'leri için dynamically allocated memory kullanıyorsak (pointerlar ile), bir destructor yazılması şart allocate edilmiş memory'i serbest bırakmalıyız o memory'i instance kullanıyor aynı zamanda bu instance yi de öldürmek istiyoruz. Default constructor bu memory'i serbest bırakıyor ardından instance'yi öldürüyor, bu yapılmasaydı bellek sızıntısı oluşacaktı.

```
110   double
111   rand
112   ()
113   {
114     /*
115      * Q:
116      *   What does the "front()" member function return for
117      *     std::vector?
118      *   There should also be a "back()" member function as
119      *     well. What does that one return?
120      * */
121     return
122       v_dist_uniform_.front()( v_generator_.front() );
123   }
```

115.satır cevabı:

"std::vector.front" ilgili vektörün ilk elemanının referansını return eder. "std::vector.back" ise ilgili vektörün en son elemanının referansını return eder.

```
125   // rand
126   std::vector<double>
127   rand_vector
128   (
129     /*
130      * Q:
131      *   Why do we declare these input arguments to
132      *     be const?
133      *   How do we assign default values for input
134      *     arguments? Could we list another input
135      *     argument below as the last one, and that
136      *     without a default value for it?
137      * */
138     const unsigned int howMany ,
139     const double       val_a = 0.0 ,
140     const double       val_b = 1.0
141   )
```

130.satır cevabı:

Const yapıyoruz çünkü gelen inputların fonksiyon içerisinde değişmesini istemiyoruz, değişmeyeceğini garanti altına alıyoruz. Burada call by value yapılmış const olmasaydı gelen inputlar fonksiyon içerisinde değiştirilebilirdi.

133.satır cevabı:

Fonksiyon () parantezleri içerisinde ilgili giriş değerine = default value şeklinde bir assign yaparsak default value vermiş oluruz. 139.ve 140.satırlarda bunu görebiliyoruz. 126.satırdaki fonksiyon 3 parametre verildiğinde de çalışacaktır sadece ilk parametreyi verdiğimizde de çalışacaktır, son iki parametre verilmez ise o iki parametre default değerleri ne ise o şekilde anlam kazanacaktır.

```
142     {
143       /*
144        * Q:
145        *   What does "assert" do in here?
146        *   This does not seem to be a good example of
147        *     the C++ way of checking for errors and
148        *     reporting them. What would you suggest
149        *     instead?
150        * */
151       assert( val_b > val_a );
```

144.satır cevabı:

Assert'in burada kullanım amacı kotrol veya hata ayıklama yapmaktır, 151.satırda val_b > val_a değil ise program sonlanacaktır, exception atılacaktır.

Burada assert beraberinde macro kullanımını getiriyor (veya assert aslında bir macrodur) bu da C++'da tavsiye edilmeyen bir şeydir fakat kesinlikle kullanılmamalıdır diyemeyiz bazı durumlarda ihtiyaç duyuyoruz.

```
152       /*
153        * Q:
154        *   Why should we avoid making use of unscoped
155        *     namespace usage calls? Below is a scoped
156        *     call, so no problem.
157        * */
```

154.satır cevabı:

Unscoped namespace kullanımı beraberinde hata getirir. Namespace tamamlanması gerekir süslü parantez kapandıktan sonra kullanıma hazır olur.

```
158       using namespace std;
159
160       /*
161        * Q:
162        *   What is constructor overloading in C++?
163        *   What does the following constructor overload
164        *     for std::vector do? Why are we making use of
165        *     it in this context?
166        *   Are there any other constructor overloads
167        *     for std::vector, which possibly would not
168        *     make sense in here?
169        * */
170       vector<double> current( howMany , 0.0 );
```

161.satır cevabı:

Constructor overloading için örneğin bir class yapısı olsun ve 2 tane private member'a sahip olsun bunların tipi de int a_ ve float b_ olsun constructor'lari şu şekilde yazılabilir;

Default cons Container() : a_(0), b_(0.0) {} , diğer constructor 1 elemanı default diğer elemanı default olmasın Container(int a) : a_(a), b_(0.0), üçüncü bir constructor ise Container(float b) : a_(0), b_(b) şeklinde yazılabilir bu şekilde C++'da constructor overloading yapılabilir. Dikkat etmek gereken bazı hususlar var onlardan birisi şudur 2 private member'da int olsaydı : operatöründen sonra gelen sıralamayı kaçırmamak lazım yoksa default olmasını istediğimiz variable'a ilk değer verebiliriz diğeri'ne default vermiş olabiliriz.

163.satır cevabı:

Vector class'ı kendi bünyesinde birkaç tane constructor'a sahiptir. 170.satırda işletilen şey current isminde bir double vektör oluştur ve bu vektörün toplam howMany tane item'ı olsun bu itemların her birine de 0.0 ilk değerini ata denmiştir.

166.satır cevabı:

std::vector<double> vec (5, ‘hello’), bu vec’i kullanarak birkaç tane constructor göstermek gerekirse;

std::vector<double> vec1(vec.begin(), vec.end()) burada iteratörler ile bir vektör kopyalaması yapılmış.

std::vector<double> vec2(vec) Direkt olarak vektörü de kopyalayabiliyoruz.

std::vector<double> vec3{‘hello’, ‘hello’, ‘hello’, ‘hello’, ‘hello’}’ de diyebiliriz.

Buradaki 4 vektör’de aynı şeyi tutar.

```
172       /*
173        * Q:
174        *   How does the range-based for below work?
175        *   Why do we work with references denoted by "vv"
176        *     below?
177        *   Does the for loop below consist of iterations
178        *     that are embarrassingly parallelizable?
179        *   If you needed to parallelize the iterations in the
180        *     loop below, then would the utilization of the
181        *     range-based for be still convenient for
182        *     some parallelization methods?
183        *
184        * */
```

Code Browser Problems

```
185       for ( auto & vv : current )
186         vv = val_a + ( val_b - val_a ) * rand();
187
```

174.satır cevabı:

Auto keyword’ü current vektörünün içindeki her bir elemanın hangi tipte olduğunu bilir, her bir item vv’ye atanır (referansı atanır kopyalama olmaz) ve her bir vv’ye bir değer ataması gerçekleşir.

Range-based ile index-based yapıları paralize olabilir 184.satıra int counter = 0 deriz ardından 187.satıra count+=1 deriz bu count üzerinden range-based loop içerisinde indexler ile de iş yapabiliriz. Tabiki burada süslü paranteze ihtiyaç olacak.

```
187
188       /*
189        * Q:
190        *   We are returning over here a variable
191        *     constructed within the scope of the
192        *     member function. Copying such an entity
193        *     into a variable in the scope of the caller
194        *     could be very costly,
195        *     since "howMany" could be really big.
196        *     However, if we are careful with the syntax
197        *     for the caller and the callee, as we are
198        *     in here indeed, then an idiom called
199        *     "Copy Elision" is implicitly invoked.
200        *     Report what this idiom is about.
201        *   Why is returning a reference to "current"
202        *     out of the question in here?
203        * */
204       return current;
205     }
206   };
207
```

189.satır cevabı:

Burada current’e en başta kaç tane item içereceği ve her bir item’ına 0 ilk değeri atanmıştı bu değer atanması olmasaydı 186.satır hata verecekti. For current içerisinde item arayacaktı ve bulamayacaktı. Peki ilk değer şart mı? Sadece std::vector<double> current ve ardından current.resize(howMany) demek yeterli olur muydu? Cevap Evet gibi duruyor. Ama for içerisinde bir kopyalama olmadığı için ilk değer (sıfırlar) verilmesi çok mühim değildir.

```
207
208   double
209   sum_of_vector_of_doubles
210   (
211     /*
212      * Q:
213      *   How do we interpret or simply read the input
214      *     argument declaration below?
215      *   Is there a particular reason for making use of
216      *     the constant reference type? Why not the plain
217      *     "std::vector<double>" type?
218      * */
219     const std::vector<double> & vec
220   )
221   {
```

212.satır cevabı:

219.satırdaki input read-only bir inputtur asla değiştirilemez ve kopyalama da yapılamaz. Bu şekilde kullanmak istememiz değiştirilmemesi ve kopyalama olmadan fonksiyon içerisinde bu input'un kullanılmasıdır.

```
221   {
222     double sum = 0.;
223     /*
224      * Q:
225      *   Could we have "for ( auto & item : ... )" below
226      *     as well?
227      * */
228     for ( auto item : vec )
229       sum += item;
230     return sum;
231   }
232
233 } } // namespace
234
```

224.satır cevabı:

Cevap evet hatta daha iyi bir yöntem olurdu. 228.satırda vec içindeki her bir iterasyonda ilgili iterasyondaki item değeri gidip item'a kopyalanıyor ve bu item üzerinden işlemler loop içinde yapılıyor. Kopyalama yapmışız toplam vec.size() kadar bir kopyalama söz konusu referansı üzerinden işimizi yapmamız daha iyi bir tercih olurdu.

```
234
235 double
236 compute_probability ();
237
238 int
239 main (void)
240 {
241   double prob = compute_probability ();
242
243   printf( "Probability of Waiting for more than 10 min:\n" );
244   printf( "  %.6f\n" , prob );
245
246   return 0;
247 }
248
249 double
250 compute_probability ()
251 {
252   using namespace std;
253   using namespace os_test::elm_218_probability;
254
255   /*
256    * Q:
257    *   What is a type alias in C++? What would be the
258    *     advantages to making use of it?
259    *   Is there any other syntax for the statement below?
260    * */
```

256.satır cevabı:

Type alias bir çok yerde kullanılıyor çok uzun bir yapıyı kısa bir isim ile kullanarak sürekli o uzun şeyi sürekli olarak kullanmamız gereken yerlerde yazmıyoruz onun yerine kendi belirlediğimiz keyword'ü yazıyoruz bunu da using keyword'ü sağlıyor.

259.satır cevabı:

Typedef keyword'ü ile de buna benzer bir yaklaşım yapabiliriz örneğin; typedef int MyInt ile using MyInt = int aynı şeylerdir.

```
261   using vector_type
262     = std::vector<double>;
263
264   /*
265    * Q:
266    *   The constructor for the indicated class does not
267    *     accept any input arguments. Then, the variable
268    *     declaration below could also be stated as
269    *     "UniformlyDistributedRVGenerator rv_gen();"
270    *     with the parentheses "()". Am I right? Is there
271    *     something wrong with the suggested call?
272    * */
273   UniformlyDistributedRVGenerator rv_gen;
274
```

265.satır cevabı:

Bu class'ın herhangi bir instance'si için ilk değer vermemize gerek yoktur, yani default cons. kullanarak instance'ler üretilecektir. Derleyici bu şekilde yani parantez kullanılmayınca oluşturulacak olan constructor'ı default cons. olarak yorumlayabiliyor. () koyup içine bir şey koymaz isek bunu normal constructor olarak algılayacaktır.

```
275   double lam = 1. / 10;
276   double time_to_wait
277     = 10.;
278   double total_min_in_hour
279     = 60.;
280
281   /*
282    * Q:
283    *   How does automatic type deduction work in C++?
284    *     Does it save the day for us in cases similar
285    *     to the declaration below?
286    * */
287   auto arrival_times
288     = rv_gen.rand_vector(NO_MC, 0., 60.);
289   vector_type result(NO_MC, 0.);
```

282.satır cevabı:

C++11 ile gelen "auto" keyword'ü sayesinde otomatik tip tespiti yapılabiliyor ve o tipte bir değişken oluşturuluyor. Örneğin bir tuple'ın ilk elemanı string olsun auto a = std::get<0>(tuple) yazarak o string'i a'ya assign edebliyoruz. Biz o tuple'ın ilk elemanının hangi tipte olduğunu bilmesek de olurdu. O tipte bir a oluşturuyor auto keyword'ü.

```
290
291   /*
292    * NOTE:
293    *   This is the actual loop fo the Monte Carlo simulation.
294    * Q:
295    *   Would the syntax "for ( int kk=0 ; ... )" compile in C?
296    * */
```

```
296    * */
297   for ( int kk=0 ; kk<NO_MC ; ++kk )
298   {
299     if ( arrival_times[kk] > total_min_in_hour - time_to_wait )
300       continue;
```

294.satır cevabı:

Cevap hayır çünkü C'de loop parantezleri içerisinde bir değişken oluşturma mümkün değildir int kk = 0 yapamayız C'de.

C'de yapılması gereken 296.satırda int kk; yazıp 297.satırda ise for(kk = 0 ; ….) yapılması gerekirdi.

```
301     else
302     {
303       /*
304        * Q:
305        *   Is "bool" a type alias or a distinct type in C++?
306        * */
307       bool flag_valid_sample = false;
308       double sample = -99.;
309
310       while ( ! flag_valid_sample )
311       {
312         double u
313           = rv_gen.rand();
314         double x
315           = - 1. / lam * log(1. - u);
316         if ( arrival_times[kk] + x < total_min_in_hour )
317         {
318           flag_valid_sample = true;
319           sample = x;
320         }
321       } // while
322
323       if ( sample > time_to_wait )
324         result[kk] = 1.;
325     } // else
326   } // for
327
328   double prob
329     = sum_of_vector_of_doubles(result)
330       / ((double) NO_MC);
331
332   return prob;
333 }
```

305.satır cevabı:

Bool sadece “true” veya “false” değerlerini alabilen bir data type’dır. Bu “true” ve “false” aynı zamanda 0 ve 1’i ifade etmektedir. Genel olarak C++’da bool nasıl kullanılır dersek;

Bool x = 0 // its a false same as bool x = false

Bool a = 100 // its a true same as bool a = true

İnt x = false + true + 6 bize 6 değerini verecektir.

Output: