GEBZE TECHNICAL UNIVERSITY

ENGINEERING FACULTY

ELECTRONICS ENGINEERING

ELEC 218

PROBABILITY AND RANDOMNESS

Project 01

Name - Surname	Abdullah Memişoğlu
Student ID	171024001

OCTAVE

Q_1:

Is there a particular reason why the "close" and "clear" statements below are stated in the order given?

'clear' komutu yanına getirilen 'keyword' e bağlı olarak sistem hafızasını değişkenler,sınıflar,scriptler veya fonksiyonlar bakımından temizlemeye yaramaktadır.Devamında gelecek kod için temiz bir çalışma alanı sağlar.

'close' komutu kod kapsamında oluşturulan figürleri temizleyen yapıdır.

Kodun ilk satırlarına yazılma sebebi ise kod içerisinde veri kaybına yol açmaması içindir.Bir değişken tanımlanıp sonrasında 'clear' komutu kullanılırsa değişken hafızadan silinecektir.

Q_2

What is a Monte Carlo type of analysis, and what is it used for ?
What do I mean by a frequency count when approximating a probability value through a Monte Carlo analysis?
What is an ensemble in Monte Carlo analysis?

Monte Carlo analizi, insanların risk analizi ve karar verme hesaplarını matematiksel dilinden bilgisayar diline uyarlayan bir simülasyondur.Bu analiz bir çok alanda aktif olarak kullanılmaktadır.Bu alanlara finans, proje yönetimi, enerji,mühendislik,araştırma ve geliştirme ve imalat sektörleri örnek gösterilir.

Çalışma prensibinden bahsedilecek olunursa , olası tüm sonuçları barındıran bir değerler aralığı belirlenir.Bu olası sonuçlarla belirli özelliklere sahip modeller inşa edilir.Sonrasında sonucu defalarca hesaplar.Her seferinde farklı modelden rastgele değerler alarak bu işlemi yapar.Modeller oluşturulurken olasılıksal dağılımlardan yararlanılır.Amaçlanan farklı sonuçlar için farklı tip olasılıksal dağılımlar kullanılabilmektedir.

'frequency count' : Monte Carlo analizinde veri setleri ile defalarca hesaplama yapıldığı biliniyor.Bunun için hesaplama sıklığı önceden belirlenmelidir.Bu sıklığa frequency count denir. Ayrıca eğer her bir durumda elde edilen sonuç not edilir ve sonuçlar elde edilme sıklıklarına göre sınıflandırılırsa bunun sonucunda oluşturulan tablo frequency plot olur.Bu terimler analiz için büyük önem arz eder.

Analiz kısmında belirli özelliklere göre ayrılmış , kalıplaşmış gruplara 'ensemble' denir.Kod içerisinde ensemble olası tüm durumları içeren gruptur.

Q3
How will each entry in "arrival_times" be distributed?

Arrival times değişkeni yapısında rand(1, no_mc) içerdiği için 60*1 ile 60*100000 arasında bi değer olacaktır ki bu da Uniformly distributed tanımına uygundur.

Q4
Explain why “result” is the variable over which the stated and relevant frequency count Operationwill be caried out.

Çünkü burada result’ın ilk değer ataması ‘0’ dır.Kodun ilerleyen kısmında if yapısı içerisinde “1” değer ataması gerçekleşmiştir.Böylece result arrayi “1” ve “0” lardan oluşacaktır.Son satırda array elemanları toplamı eleman sayısına bölünerek olasılık hesabı yapılacaktır.

Q5

Recall what the numel() function does.How are the iterates in this for loop indexed ?

numel() fonksiyonu argüman olarak array alır.Fonksiyon array argümanının eleman sayısını döndürür.İsmi “number of elements” den gelir.Kod içerisinde for döngüsü (for kk = 1:numel(arrivel_times)) kk değişkeni 1’den arrival_times arrayinin eleman sayısına kadar 1’er artacaktır.

Q6
What does “continue” do ? Could the following “if else” control flow statement be constructed without resorting to “continue”?

“continue” komutu for veya while loop içerisinde kullanılarak bir sonraki bir sonraki iterasyon adımına geçişi sağlamaktadır.Bir döngü içerisinde belirli if else yapıları kurulduğunda bazı durumlarda işlem yapılmaması gerekir.Bu durumlarda döngünün bir sonraki iterasyona geçişini sağlayan yapıdır.Burada “continue” kullanmamak adına if(A)➔continue , else➔ (gerekli işlem) olduğunu düşünürsek if(~A)➔(gerekli işlem) denilerek else komutu kullanılmadan sorun çözülebilir.

Q8
Would the code below work correctly with flag_valid_sample set initially to 0 (zero)? What does logical() signify? Is it a function call or something else? Is there a chance for "sample" to continue having the assigned value "[]" at the end of the "else" statement? What does "[]" mean in Matlab/Octave? Will the two variables continue to remain in scope when we are done with this "else" part of the "if else" control flow statement?

Logical() komutunun tanımı şöyledir.
X = logical(A) ise ve A’nın elemanları ≠ 0 olduğu sürece X=logical 1 , aksi halde X = logical 0 olacaktır.Buradan flag_valid_sample değişkeninin atamasının “0” olduğu görülebilmektedir.logical() dahili bir fonksiyondur. “[ ]” boş bir numerik matris oluşturmaya yaramaktadır.x = [ ] ise x boş numerik bir matristir.”else” içerisinde değer ataması yapılan bu değişkenler “else” yapısından çıktıktan sonra da değerlerini koruyacaktır.Değişkenin sıfırlanması olayı fonksiyonlarda gerçekleşmektedir. Fonksiyon içerisinde atama yapılırsa değişken sadece fonksiyonun içinde o değeri alır.Bu iki değişken “else” içerisinde aldığı değerleri korur.

Q9
Could the while loop below continue to run through iterations indefinitely, is there such a Possibility?

Evet eğer while döngüsü içerisinde if yapısı gerçekleşmezse while döngüsü sonlanamayacaktır.if yapısı gerçekleşmelidir.

Q10
If your answer is yes to the previous question, then how can we expect the while loop below to be finalized in a finite amount of time? What is the answer to this question founded upon?

for döngüsü içerisindeki else yapısı gerçekleşirse while döngüsü gerçekleşir.Bu durumda for döngüsü else yapısı incelenmelidir.else yapısının gerçeklenebilmesi için
arrival_times(kk) < total_min_in_hour - time_to_wait
olmalıdır.while döngüsü içerisindeki if koşulunun gerçeklenmesi ise şöyledir;
arrival_times(kk) + x < total_min_in_hour
bu durumda u=rand() ile hesaplanan x değişkeninin time_to_wait değişkeninden küçük olması durumunda while döngüsü sonlanacaktır.
x<time_to_wait sağlanmalıdır.

Q11
How is the inverse transform method utilized for the generation of a sample "x" obeying the exponential distribution with parameter lambda ("lam" in the code)?

$$x = \frac{-1 * \log(1-u)}{lam} ==> -lam * x = \log(1-u) = [\exp(.)] =>$$
$\exp(-lam * x) = 1 - u ==> u = 1 - \exp(-lam * x) = cdf_x(x)$ for exponential RV.

Q12

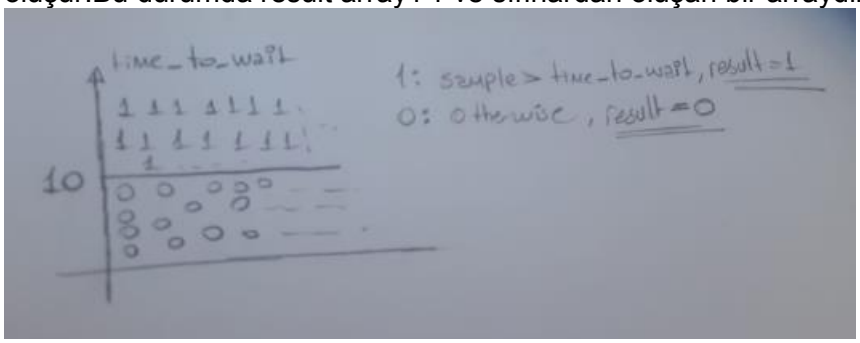
Is it the sample of an exponentially distributed random variable that we need when "x" has to be generated? If no, then what do we need? Explain how the "if" control flow statement below implements the rejection method necessary to generate samples of a random variable belonging to a truncated distribution. Is it below that there happens to be a statement that is able to terminate the enclosing "while" loop?

Rejection Method: Bu method ile belirli şartları sağlayan bölgedeki örneklemler seçilerek işlemler o örneklem kümesinde yapılır.Burada x değeri hesaplanır ancak arrival_times(kk) + x > 60 ise sample = x değer ataması gerçekleşmez.Bu durumda rejection kavramı gerçekleşir.Bir önceki soruda cdf değerinin exponential RV'a uygun olduğu gözlemlendi.Buna göre if yapısı ile sınırlı bölgede exponential RV hesabı Truncated distribution tanımını sağlar.

Q13

"result" is our record-keeping variable for frequency count. We have a conditional entry setting with assignment to 1. What does this signify? What would assignment to 0 signify and why do we not have it explicitly coded below?

Result değişkeninin ilk değer ataması "0" olarak gerçekleştirildi.
Result = zeros(1,no_mc);
Bu durumda yeni bir atama yapılmadığı sürece result değeri "0" olacaktır.
Eğer if sample > time_to_wait koşulu sağlanırsa result(kk) = 1 değer ataması oluşur.Bu durumda result array'i 1 ve sıfırlardan oluşan bir arraydir.



Q14

This is where the frequency count operation through the record- keeper "result" is done. Comment what would happen if we chose have a "result" variable with many more entries. Why cannot the operation below yield a value greater than 1 or smaller than 0? Why is the operation below yield an approximation for the requested probability?

Sonuç olasılık değeri sum(result) / numel(result) olarak verilmiştir.Bu olasılık değeri 1'den büyük ve 0'dan küçük değer alamaz.Çünkü bir önceki soruda "result" değişkeninin sadece 0 ve 1 değerlerini aldığı belirtilmektedir.Ayrıca **sum(result)/numel(result)** operasyonu aritmetik ortalama işlemidir.Aritmetik ortalamanın kuralı ortalama sonucu ortalaması alınan sayıların en küçüğünden küçük , en büyüğünden büyük olamaz.Bu kurala göre sonucun 0 dan küçük , 1'den büyük olmaması beklenir.Daha fazla 'result' girdisi daha fazla örneklem demektir. Final bölümünde de görüldüğü üzere bu bir yaklaşım algoritmasıdır.Daha fazla 'result' girdisi daha iyi yaklaşım demektir.

Final:

```
>> probability_octave_project
Probability of Waiting for more than 10 min:
  0.259260
>> probability_octave_project
Probability of Waiting for more than 10 min:
  0.257710
```

Kod çalıştırıldığında farklı değerler aldığı görülmektedir.Kodun amacı yakınsamadır.Burada değerler rand()➔ rastgele seçilmektedir.Belirli bir veri olmadığı için amaç beklenen değere yakınsamasıdır.

PYTHON

Q_1

What is an import alias? What would we do to import just a single submodule inside a module with an alias? Is it possible to import multiple modules with a single import command?

Import Alias: Bu terim python'da bir fonksiyonu takma isim ile tanımlama işlemidir.Kod içerisinde **import numpy as np** satırında numpy mdoülü import edilir ve bu modülün kod içerisinde **np** anahtar sözcüğü ile çağırılacağı belirtilir. Bir nevi kodlamada hız kazanabilmek amacıyla kısa anahtar sözcük ataması denilebilir.Ayrıca birden fazla modülü tek bir satırda import etmek de mümkündür. Örnek durum aşağıdaki gibidir.

```
"""
x.py
"""
import numpy
import matplotlib
```

```
"""
y.py
"""
From b import *
```

Yukarıda belirtilen 2 adet kod parçasında x.py import edilecek modülleri,y.py ise x.py kodu içinde import edilen tüm modülleri tek bir satı ile import etmektedir.

Q_2

What does the command above do? If I did not have this one in the octave implementation, how must I have decided that a seed determination is not necessary to be carried out by the developer? What is the default seed for the random number generator when called as above? Why would we refrain from feding a seed with a constant value to the generator? Also what does the above call tell us about the seed value determined being global or local? What may be the advantages or disadvantages associated with either alternative (i.e., global or local seed value)?

rnd.seed() "rastgele sayı oluşturucu"(Random number generator) görevi üstlenmektedir. seed(a), değeri bir sonraki adım için üretilen random sayı değerini tutmaktadır.seed(a) kullanılmadığı sürece kod her çalıştığında farklı random değerler oluşturulacaktır.Bu yüzden rnd.seed() içerisine bir input girilmemiştir.Çünkü her seferinde yeni bir random number üretilmek istenmektedir.Eğer bunu octave'da yapmak isteseydik rng(seed) komutu kullanılabilirdik.Ayrıca bu komut ile üretilen tüm değerlerin 0-1 aralığında sayı üreteceği bilinmektedir.

Q3
What do we do for line continuation for a single command? We have examples below for assignments.

Satırdaki işlemi bitirmeden işleme diğer satırda devam edebilmek için **“ \ ”** işareti kullanılır.

Q4
What is the list data structure in python? What is a generator (I am not referring to the random number generator in here) in python? How will the random variable samples generated with "rnd.random()" below be distributed? What will be probabilistic distribution associated? And what happens when we multiply the samples generated by "rnd.random()" with a constant, as in the python generator below? How will the resulting samples be distributed now?

Listeler , Python’da art arda sıralanan verileri tutan yapılardır.Listelerin her bir elemanına öğe(item) denir.Kullanım amacı farklı veri türlerini bünyesinde barındıran bir veri topluluğu oluşturmak.Bu listelerde her öğenin kendine ait indeksi bulunur.

```
"""
arrival_times = \
  [ total_min_in_hour*rnd.random() for kk in range(no_mc) ]
"""
```

Bu kod satırında arrival_times bir liste belirtir.Bu listenin her bir elemanı for döngüsü ile o adımdaki rnd.random() sayısına bağlı olarak yeni değer alır.Bu değerler ile no_mc elemanlı bir arrival_times listesi oluşturulur.rnd.random() for döngüsündeki her iterasyonda yeni bir değer alır.Çünkü rnd.seed(a) tarzında random sayının indeksini kullanmadık.Her defasında yeni bir random sayı üretilir.Üretilen sayı için rnd.random() ile üretilen her sayı 0-1 aralığında değer alacağından arrival_times listesinin tüm elemanları (60*0,60*1)=(0,60) arasında değer alır.Böylece sonucun Uniformly distributed olduğu gözlenir.

Q5
What is the return type of "np.zeros()", as used below? Is it one that corresponds directly to the matrix data structure of octave? Or is it something else? What are the dimensions for "result" below going to be? How can we check the dimensions through the "shape" attribute? And what is the type for the "shape" attribute of "result"?

np.zeros() fonksiyonu 3 adet girdi alır.Bunlardan ikisi opsiyonaldir.aldığı ilk girdi oluşturulacak array in boyutunu belirler.2.si type belirler ancak 2. Girdi opsiyonaldir.Eğer kullanıcı tek bir girdi kullanırsa tipi default olarak float atanır.Yani fonksiyonun return tipi float olur. np.zeros(a) nın çıktısı a elemanlı sıfırlardan oluşan bir array dir.Bu fonksiyonun octave karşılığı çok benzerdir.**X=zeros(sz)** olarak belirlenir.Burada sz matrisin satır sütun sayılarını yani array büyüklüğünü belirtir.

```
result = \
  np.zeros(no_mc)
```

Burada result no_mc elemanlı sıfırlardan oluşan bir arraydir.

Q6
What is the return type for each of "range()" and "len()"? Could not I have utilized "np.arange()" instead of "range()"? Are the return types for the stated functions the same? If not how can they be used interchangably in the statement below? What does "range()" return with a single input argument? Are there other possibilities for the input argument set of "range()"? Could I iterate through "arrival_times" in a different way, possibly making use of the "in" keyword? Why do I not prefer the indicated alternative in here?

range() fonksiyonunun syntax'ı şöyledir.range(start,stop,step) burada start ve step opsiyonaldir.Eğer kullanıcı tarafından girilmezse start default olarak "0" , step ise "1" olarak alınır.range fonksiyonu çıktı olarak start'tan stop'a kadar step artış ile oluşturulmuş bir liste döndürür.range fonksiyonunun return tipi listedir.Python 3'te ise bu return tipi değişmiş ve 'iterator' olmuştur.
np.arange() fonksiyonu range() ile aynı syntax'a sahip aynı işlevi gören numpy modülüne sahip yapıdır.Aralarındaki fark np.arange() fonksiyonunun return tipi array olmasıdır.Ayrıca np.arange() fonksiyonu çok büyük değerler için daha hızlı çalışmaktadır.
Burada np.arange() ile range() birbirleri yerine yazıldığında kod yine çalışacaktır.Bunun sebebi built-in len() fonksiyonu listelerin eleman sayısını ölçmekle beraber numpy için kullanıldığında arrayin 1. Boyutunun eleman sayısını döndürür.Burada array zaten 1 boyutlu olduğu için birbiri arasında değişim sağlanabilmektedir.
'len()' fonksiyonunun return tipi ise integer'dır.Girdi olarak verilen listenin eleman sayısını belirtir.
'in' ile length hesaplama şöyle olacaktır.

```
counter =0;

for i in arrivel_times:

        counter += 1
```

Q7
What is the common type for the "True" and "False" values in python?
bool type

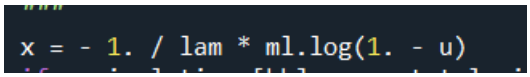
Q8
Does it matter what "sample" is assigned to at this point? What is special about the "None" type? Are there any contexts where the usage of "None" perhaps makes more sense?

None tipi özel bir durumdur.$None \neq False$, 0 . None "null" bir değişken tanımlamak amacıyla kullanılan bir anahtar sözcüktür.Python'da None ,NoneType sınıfına ait bir data tipidir.None ile kıyaslanan herşeyin sonucu False olur .Sadece kendisi ile kıyaslandığında True değeri alır.

Q9

What is the difference between "1" and "1." in python? Would the value of "x" below differ if I had used "1" instead of "1." in the indicated assignment? How is the call for the "log()" function different compared to the call in octave? Are there any more functions encapulated inside the "math" module?

1 ve 1. arasındaki fark veri tipi ile ilgilidir.



Burada eğer 1. yerine 1 kullanılsaydı sonucun veri tipi integer olurdu. 1. Kullanıldığında devamındaki değişkenin veri tipi ne ise sonucun veri tipi de o olacaktır.

log() fonksiyonu math modülünde bulunan bir fonksiyondur.bu fonksiyonun syntax'ı log(x[,base]) şeklindedir.Burada eğer base belirtilmezse (Kod içerisinde belirtilmemiş) default olarak doğal logaritma atanır. Buna ek olarak math modülünde ;

ceil(x): x bir float sayı x'ten küçük veya eşit olan en küçük integer sayıyı döndürür.

copysign(x,y): x değerinin başına y'nin işaretini ekler ve sonucu döndürür.

fabs(x): x'in mutlak değerini alır.

factorial(x) : x'in faktöriyelini alır.

exp(x) : exp(x) döndürür.

log10(x) : tabanı 10 olan logaritma değerini döndürür.

pow(x,y) : x^y döndürür.

Sqrt(x) : Kök alır.

sin(x) , cos(x) , tan(x) ….. daha bir çok fonksiyonu içerisinde barındırmaktadır.

Q10

Notice that I did not have to code a function to sum up the entries in "result", how is that possible? What is "np.sum()" able to do in order to compute the stated sum? If you have a type that you defined yourself in python, perhaps through a class definition of yours, how would you enable the "len()" function to operate on it? Checking the member functions for the type that "result" belongs to would help in this case.

np.sum () fonksiyonu bir numpy arrayindeki elemanları toplamak için kullanılan numpy modülü fonksiyonudur.result değişkeni bir numpy arrayi olarak olarak tanımlanmaktadır.Bu durumda np.sum() fonksiyonu ile içeriğindeki tüm öğeler toplanabilmektedir.
Eğer kendim bir fonksiyon oluşturup toplamı bulsaydım result değişkeni numpy modülü kullanmadığımdan liste olacaktır.Listelerin öğe sayısını bulmak için len () fonksiyonu kullanılır.Eğer numpy kullanılacaksa array in tek boyutlu olmasına dikkat edilmelidir.Çünkü birden fazla boyutlu arraylarde len() fonksiyonu sadece 1 boyuttaki eleman sayısını hesaplayacaktır.Böylece sonuç hatalı çıkacaktır.

FİNAL:

```
Probability of Waiting for more than 10 min:
  0.257490

In [37]: runfile('E:/ELM218_Code_Project/
codes/content/
truncated_exp_rv_samples_Monte_Carlo_octave_
python_c_cpp/python/prob_waits_for_more_th',
wdir='E:/ELM218_Code_Project/codes/content/
truncated_exp_rv_samples_Monte_Carlo_octave_
python_c_cpp/python')
Probability of Waiting for more than 10 min:
  0.256020
```

Kod çalıştırıldığında doğru şekilde çalıştığı görülmektedir.Octave kodunda gördüğümüz değere tekrar yakınsadığı gözlendi.Ve yine bu bir yakınsama olduğundan random değerlere göre değişim gösterecektir.

#C++ KODU

Q_1

What is the advantage of having the opportunity to make use of namespaces or even nested namespaces?

Namespace'ler aynı isimde fonksiyonların çakışmasını önlemek için oluşturulmuş yapılardır.Örneğin bir xy() fonksiyonu tanımladık ve kod içerisinde kullandık.Bu fonksiyon ile compiler kapsamında aynı isimdeki bir fonksiyon olduğunu düşünelim namespace bu karmaşayı önleyecektir.Bir nevi dökümantasyon sağladığı söylenebilir.

Q_2

Do we need this "private:" indicator? Can we do without it?

Private satırı silindiğinde kodun tekrardan doğru çalıştığı görüldü.public bir değer olsa da çalışmaktadır.Ancak private değerlere sınıf dışından erişimi engelleyeceğinden güvenlik açısından gereklidir.

Q_3

What are the advantages and disadvantages to keeping the generator and the distribution in vectors as indicated above? As a hint, one advantage will become clear in the constructor.

Vektörler c++ içerisinde dinamik dizilerdir.Bunun anlamı şudur.Diziler tanımlanırken eleman sayısı baştan belli olmalıdır.Vektörün avantajı eleman sayısını işlem sırasında değiştirebilmesi olmaktadır.Dezavantajı ise bu durumu sağlayabilmesi için hafızada yerinin sürekli tazelenmesi gerekir.Bu yüzden de hafızada fazla yer kaplayacaktır.

Q4

What is the "epoch" in this context? Is the seed thus set through the current time information a local or global one? What would be advantages to making use of a local seed value? Should the seed for a random number generator be set every time a sample obeying a certain distribution is created? Or is this a gruesome mistake? Is this how we do it in the current source code?

epoch tarih ve zaman orjini olarak alınan bir noktadır.Kod içerisinde zamana bağlı bir şey var ise o epoch üzerinden geçen tik sayısına bağlıdır.Kod üzerindeki 30 saniye epoch üzerinden 30 tik geçmiştir.Burada seed bir sınıf içerisinde oluşturulduğundan local'dir.Local bir değişken tanımlamanın yararı daha düzenli ve güvenilir bir yapı sağlar.Bunun yanında kod çalıştığı sürece var olmazlar.Sınıf çalışmayı durdurduğunda hafızadan silinirler.Böylece kapsamlı kodlamalarda hafızayı yormaz.seed değeri her defasında farklı bir değer almalıdır.Bunu kod çalıştığında sürekli farklı sonuç aldığımızda görürüz.Aynı zamanda seed değerinin gerçek zamana bağlı değiştiği gözlenmektedir.Bu da her defasında random sayı üretmesini sağlayacaktır.

Q5

What are the arguments to the "emplace_back(...)" member function of a vector? Do the number and types of the indicated input arguments differ for different template parameters for std::vector? "emplace_back(...)" makes use of a paradigm called perfect forwarding. What is this concept? Do the "emplace_back(...)" calls below involve object copying operations? If yes, is there an extra call we may utilize to prevent copy operations during intended in-place constructions in a vector?

Emplace_back(..) fonksiyonu bir vektörün sonuna ekleme yapmaya yaramaktadır. Burada fonksiyon için girdi değeri vector.emplace_back() şeklinde yazıldığında vector değişkeninin sonuna öğe olarak eklenecektir.Burada fonksiyonun girdisi eklenecek vektörün elemanlarıyla aynı tipe sahip olmalıdır.Çıkarımla her vektör için girdinin tipi değişecektir.Bunun yanında **perfect forwarding** tanımı şöyledir.Dizinin elemanlarını değiştirmeden ve kopyalamaya gerek duymadan vektör sonuna eklenecek öğelerin iletimine denir.

Q6

What does the following syntax mean for the destructor? Are there any other reserved keywords that could be used instead of "default"? Why is "default" the one that makes sense in here?

Burada default "Pure virtual" a uygun destructor sağlar.Böylelikle işlevi biten her yok edilir.Ayrıca

```
~UniformlyDistributedRVGenerator()
  = default;
```

Kod parçasında default➔{ } yazarak da oluşturulabilmektedir.

Q_7

What does the "front()" member function return for std::vector? There should also be a "back()" member function as well. What does that one return?

Front() fonksiyonu başına gelen vektörün ilk indeksteki elemanını , back() ise son indeksteki elemanını döndürür.

Q_8

Why do we declare these input arguments to be const? How do we assign default values for input arguments? Could we list another input argument below as the last one, and that without a default value for it?

Burada val_a ve val_b değerleri her zaman 1 ve 0 olmalı değişmemelidir.Çünkü algoritma 1 ve 0 örneklemlerini bir vektörde saklayıp en son ortalama alarak bulmak için yazılmıştır.Bu durumda val_a ve val_b değerlerinin 0 ve 1 olması gereklidir.

Q_9

What does "assert" do in here? This does not seem to be a good example of the C++ way of checking for errors and reporting them. What would you suggest instead?

Assert fonksiyonu “conditional expression” ile handling sağlar.Eğer assert(….) içerisi true ise kod çalışmaaya devam eder.Eğer false ise kod hata verecektir.Buna alternatif iyi bir yöntem exception handling’dir.Try ve catch yapısı ile kontrol sağlanabilir.

Q_{10}

Why should we avoid making use of unscoped namespace usage calls? Below is a scope call, so no problem.

Unscoped namespace’lerde fonksiyon tip ve değişkenlerde bir anlam karmaşası olacaktır.Değişkenlerin tekrar etmemesi takip edilmelidir.Bu yüzden verimli bir kullanım değildir.

Q_{11}

What is constructor overloading in C++? What does the following constructor overload for std::vector do? Why are we making use of it in this context? Are there any other constructor overloads for std::vector, which possibly would not make sense in here?

Constructor overloading:Aynı constructor ismi ile girdi sayısı değiştiği sürece birden fazla tanım yapılabilmesi durumudur.Overload edilmiş constructor 2 argümanlı yeni constructor yapısıdır.Burada constructor girdisi 0 ve howMany olacaktır.

$Q_{12:}$

How does the range-based for below work? Why do we work with references denoted by "vv" below? Does the for loop below consist of iterations that are embarrassingly parallelizable? If you needed to parallelize the iterations in the loop below, then would the utilization of the range-based for be still convenient for some parallelization methods?

Range-based for döngüsü düzenli veya düzensiz artışa sahip bir vektörün tüm elemanlarını dolaşmaya yaramaktadır.Örnek olarak

```
vector<int> v = {1, 3, 5, 7, 9};

for (auto x : v)
    cout << x << ' ';
```

Burada çıkış v vektörü olacaktır.

Vv referansı ise burada vektör değerleri değil adresleri gezileceğinden & kullanarak vv referans değerleri gezildi.Bir data dizisi üzerinde iterasyonlar yapılır.Paralelizm vektör üzerinde gezerken her elemana sırayla bakmak yerine tüm elemanlara neredeyse aynı zamanda bakmaya çalışmaktır.Burada range based for döngüsü paralelizm için uygun değildir.Paralelizm for içinde sürekli artış için i nin geçmişteki değerine bağlıdır.Ancak range based for döngüsü düzensiz artış gösterir.Bu durumda 1 birim gerideki değerl sürekli artış gösteren for döngüsü için doğrulanır.Paralelizme uygun olan sürekli artışa sahip for döngüsüdür.

$Q_{13:}$

We are returning over here a variable constructed within the scope of the member function. Copying such an entity into a variable in the scope of the caller could be very costly, since "howMany" could be really big. However, if we are careful with the syntax for the caller and the callee, as we are in here indeed, then an idiom called "Copy Elision" is implicitly invoked. Report what this idiom is about. Why is returning a reference to "current" out of the question in here?

Copy Elision: Copy Elision bir derleyici optimizasyon tekniğidir.Gereksiz obje kopyalamayı elimine etmeye yarar.Burada değer kopyalamamak adına call by reference yani adresten alma durumu söz konusu olacaktır.

$Q_{14:}$

How do we interpret or simply read the input argument declaration below? Is there a particular reason for making use of the constant reference type? Why not the plain "std::vector<double>" type?

Q15:

Could we have "for (auto & item : ...)" below as well?

Koda & eklenerek çalıştığı gözlemlendi.

Q16

Q: What is a type alias in C++? What would be the advantages to making use of it? Is there any other syntax for the statement below?

Alias : Türleri kullanıcının kendi kullanacağı ismi vermesidir.

Avantajı kullanıcı tarafından verilen isimler ile kodun yazma ve okuma kısmında sorun yaşanmamasıdır.

```
 * */
using vector_type
  = std::vector<double>;
```

Bu satira alternatif olarak aşağıdaki gibi yazılabilir.

typedef std::vector <double> vector_type;

Q17

The constructor for the indicated class does not accept any input arguments. Then, the variable declaration below could also be stated "UniformlyDistributedRVGenerator rv_gen();" with the parentheses "()". Am I right? Is there something wrong with the suggested call?

Bu şekilde tanımlama yapılamaz.Çünkü () koyarak null bir input çağırılmış olur. İndicated class constructor hiçbir şekilde girdi almaz.

Q18

How does automatic type deduction work in C++? Does it save the day for us in cases similarto the declaration below?

Auto kullanıldığında : Türün int, double , float olduğu belli olmadığı durumda atama sonucunda eşitliğin sağ tarafı türü ne ise derleyici buna göre tür belirler

Bu tanımlama çok fazla yerde verimli bir şekilde kullanılabilir.

Q19

Would the syntax "for (int kk=0 ; ...)" compile in C?

Bu syntax'ta for içerisinde int bulunması C'nin compiler standartlarına bağlı olarak değişmektedir.C99 ve üzeri sürümlerde derleyecektir.

Q_{20}

Is "bool" a type alias or a distinct type in C++?

Distinct tipidir.

Before the **bool** type was introduced to the C++ language, we frequently saw a declaration of the following form in C++ programs:

```
enum Boolean { FALSE, TRUE };
```

which makes Boolean a distinct user-defined type with literals (constants values) TRUE and FALSE. However, C++ now has an integral type called **bool** with literals **true** and **false**.

FİNAL:

```
Probability of Waiting for more than 10 min:
  0.260010
```

Kodun doğru şekilde yakınsadığı kıyaslanarak gözlemlendi.

C KODU

Q_1

Is there any reason to refrain from using a "define" statement as below in C or C++?

Define ile tanımlanan değişkenler kod içerisinde hiçbir zaman değişmeyecek olan değişkenlerdir.Yeni değer ataması yapılamaz.İlgili kodda no_mc için yeni bir değer atamasına ihtiyaç olmadığından kullanmanın bir sakıncası yoktur.

Define ile tanımlamaya preprocessor tanımlama denir.

Q_2

What does the call to "rand()" return below? And why do we divide it by "RAND_MAX"? What would happen if we had not resorted to C-style type-casting through "(double) ..."?

rand() fonksiyonu C dilinde 0 – RAND_MAX arasında rastgele bir sayı üretir. (RAND_MAX = 32767).Üretilen random sayının 0 – 1 aralığında kalması istendiği için üretilen random sayı RAND_MAX'e bölünmüştür.rand() fonksiyonu integer sayı üretir.RAND_MAX sayısı da integer bir sayıdır.Eğer (double) keywordü eklenmezse sonuç int olur sonucun her zaman 0 olması beklenir.0.xxx olması için (double) keywordu eklenir.

Q_3

Are they any benefits to making use of an input argument with type "double **"?

Yararı vardır.Burada bir pointer oluşturulmaktadır.Sonrasında memory allocation için kullanılacaktır.Memory allocation array boyutunu kod çalıştığı sürece dinamik bir şekilde değiştirebilmeyi sağlamaktadır.

Q_4

What is the difference between "calloc(...)" and "malloc(...)" for memory allocation? Except possibly the syntactic one, involving the number and types of input arguments? Will we explicitly make use of an advantage (perhaps an extra operation) that "calloc(...)" provides us?

Malloc() fonksiyonu tek bir input alır ve saklaması gereken öğe sayısını type a bağlı byte değeri ile çarpıp hafızada büyük bir blok oluşturur.Sonunda fonksiyon bu büyük bloğun ilk değer pointer'ını döndürür.

Calloc() ise 2 input alır.Biri hafızada tutacağı öğe sayısı diğeri ise tipine bağlı olarak değişen size ıdır.Hafızada her öğe için bir blok ayrılacak şekilde oluşturulur.

Malloc () fonksiyonu calloc() dan daha hızlıdır.Ancak Malloc() fonksiyonu overflow a yol açabilir.Bu durumda calloc() daha güvenilirdir.

Ayrıca calloc() ilk değer atamasını "0" yaparken malloc "garbage value" barındırır.

Q_6

Would the "for (int kk=0 ; ...)" syntax work in C? Does the assignment "ptr_alias[kk] = ..." point out the advantage of using the alias? Why would one prefer in the for loop below "for (... ; ++k)" instead of "for (... ; k++)"?

C nin yeni nesil compiler'ında bu syntax çalışmaktadır.C99'dan sonraki versiyonlarda çalıştığı görülmektedir.

++kk ve kk++ arasında fark vardır.İlki değeri arttırır ve o değer ile işleme girer diğeri ise işleme girer sonrasında değer artar.Ancak for loop içerisinde ++kk ve kk++ arasında çalışma anlamında hiçbir fark yoktur.Performans ve optimizasyon açısından ++k kullanılmasının avantajlı olduğu bilinmektedir.

Q_7

What is the reason for the scale and shift operations above, for each of the samples returned by the function"get_fp_random_standard_uniformly_distributed_single()"?

Burada yapılan işlem şöyle açıklanabilir.Burada Uniformly distributed RV kullanılır.cdf fonksiyonu incelendiğinde $\boldsymbol{cdf_x(x) = \frac{x-a}{b-a}}$ olduğu bilinmektedir.Burada x yalnız bırakıldığında

$\boldsymbol{cdf_x(x) * (b-a) + a = x}$ bu eşitlik scale ve shift kullanımını açıklamaktadır.

Q_9

Do we need this prototype? Can we do without it?

C dilinde definition – declaration tanımına göre eğer bir fonksiyon bir satırda kullanılacaksa o satırdan önceki satırda ya definition ya declaration yapılmış olmalı Fonksiyonun tanımlaması kod sonunda bulunacaksa başta deklare edilmeli. Deklarasyon olmadan da yapılabilirdi.Fonksiyon kullanılmadan önce başta tanımlanırsa deklarasyona gerek yoktur.

Q_{10}

Why would "%8.6f" and similar ones to that with also almost any integer above "8" make sense for the formatting call above, but not "%7.6f" and any integer below "7"?

Algoritma virgülden sonra 6 basamak için yaklaşım uygulamaktadır.Bu durumda 7-digit çalışmaktadır.Bu yüzden %8.6 ve üzeri için doğru yaklaşım elde edilebilir.

Q_{11}

How do the following two lines of code set the seed for the random number generator, using the current time information? Is the typecast "((unsigned) ...)" necessary below?

Srand() her çağrıldığında farklı random sayı üreten bir fonksiyondur.time_t ise o anın tarih-saat bilgisini taşıyan yapıdır.srand() fonksiyonu yapısı gereği unsigned girdi alır. Bu sebepten girdinin unsigned tipinde olduğundan emin olunmalıdır.Unsigned değil ise (unsigned) kullanılarak çevrilmiştir.Bu sebepten (unsigned) kullanımı gereklidir.

Q_{12}

If, in the function call below, what we are sending as the 1st argument is effectively of type "double **", then why not declare a variable through "double ** arrival_times_ptr_ptr;" and send this one instead as the indicated argument? Would the code compile? Would the code work?

Double** kullanarak arrival_times_ptr adresine değil o adres değerinin bulunduğu adrese ulaşılmaktadır.Bu durumda 2 adım sonra & arrival_times_ptr a ** ile ulaşılamadığından double** ile bu durum sağlanamaz.

Q_{13}

Why do we particularly prefer "calloc(...)" instead of "malloc(...)" in the call below? As a note, the array pointed to by "result_ptr" is designed to hold "0." or "1." in its entries.

Calloc ,malloc() tan farklı olarak arrayleri her bir öğeyi ayrı saklamaktadır.malloc() bir adet büyük blok oluşturur ve her öğeyi orada saklar.Burada tercih edilme sebebi ise malloc içeriğinde “garbage value” barındırır.calloc ise ilk değer atamasını her zaman “0” olarak ayarlar.

Q_{14}

Is this a so-called embarrassingly parallelizable loop? If yes, look up methods to parallelize the iterations in the loop.

Evet burada paralelizm sağlanabilir.Bir önceki sorularda range-based for döngüsü için paralelizm olmayacağını çünkü düzenli bir artış olmayışından bahsedilmişti. Burada düzenli bir artış bulunmaktadır.Böylece i-1 ile arrayin bir önceki indeksteki öğeye erişebilme olanağı vardır.Bu şartı sağlaması paralelizm olabileceği anlamına gelmektedir.

Q_{15}

Is "bool" only a type alias or a distinct type in C? If the former is the correct way to describe "bool", then how are "false" and "true" defined in C?

Bool c dilinde alias type’dır.<stdbool.h> header dosyası bulundurur.Burada bool ➔ _Bool olarak tanımlanır.TRUE değeri C dilinde !FALSE olarak tanımlanmıştır. Bool C diline girmeden önceki kullanıma uygun bool tasarlanmıştır.Tasarıma göre FALSE ➔ “0” bununla birlikte “0” harici her karakter TRUE olarak değerlendirilir.

Q_{16}

Will "sample" ever continue to maintain the indicate value set through initialization after we manage to break out of the while loop?

Sample değişkeni bir fonksiyon içinde tanımlandığından fonksiyon çalışmayı bitirdiği an bünyesinde tanımlanan tüm değişkenler hafızadan silinecektir.

Q_{17}

Why is there not an else statement above setting the indicated entry to 0.? Why can we do without it?

Else yapısına gerek yoktur.Çünkü result_ptr değişkeni calloc ile oluşturuldu.calloc() calloc() fonksiyonu malloc()’tan farklı olarak bünyesindeki değişkenin ilk değer atamalarını ‘0’ olarak ayarlar.Bu durumda zaten result_ptr arrayi ‘0’ larla dolu bir array if yapısına girdiğinde ‘1’ değeri alacak girmediğinde else ‘e gerek kalmadan ilk değeri olan ‘0’ değerini alacaktır.

Q_{18}

Do we actually need the typecast "((double) NO_MC)" above? Can we do without it?

Evet double kullanılmalıdır.Çünkü burada result_ptr unsigned int veritipinde bir değişkendir. Buna göre sonucun unsigned int olarak sürekli '0' gelmemesi adına (double) eklenir.

Q_{19}

What would happen without the "free(...)" statements above?

Free(..) komutu kullanılmadığı takdirde hafızada değişkenler için ayrılan alan kod çalışmadığında dahi yer tutacaktır.Hafızadan silinmeyecektir.Bu da hafızada gereksiz yer kaplamasına yol açacak ve sistemi yavaşlatacaktır.

FİNAL:

```
Probability of Waiting for more than 10 min:
     0.259410
```

Kod birden fazla kez çalıştırılarak ve 4 kod kıyaslanarak doğru yaklaşım değerleri gözlemlenmiştir.