Problem Statement

In this project, we need to simulate model of bank queues. The customer arrival process is Poisson with rate λ. The 75% of customers have simple transactions to do and 25% have complex transactions. The service time for simple transaction is Erlang-2 distribution with mean 2 minutes and that of complex transaction is Erlang-5 distribution with mean 6 minutes.

We need to simulate 3 different strategies as below:

A. We have only single queue for 3 tellers.
B. Each teller has a separate queue. Customer joins the shortest queue.
C. We have two queues, one for simple transactions and other for complex transactions. Relative teller will handle each queue.

Part A

Code:

```
clc
clear all
close all
lambda=1/2; % arrival rate per second (1 minute per packet)
T=1*3600; % simulation time in second (10 hours)
delta=0.1; % simulation step size in second
simulationTime=T/delta; % number of simulation steps
currentSlot=0; % Simulation time
noOfcustomers=0; % Total number of customers
tellerStatus=zeros(3,1); % Tells whether teller is active or not
activeBuffer=zeros(3,1); % Tells which customer is currently being served
oldBuffer=zeros(3,1);
customerId=[]; % Stores the customer IDs
customerServed=0; % Number of customer served
queueLength=0; % Tells the queue length at each time slot

muSimple=2/60; % mean for simple transactions
muComplex=6/60; % mean for complex transactions
count = 0;
while currentSlot < simulationTime
    currentSlot=currentSlot+1;

    % Check if new customer is arriving
    R=rand(1);
    if R<lambda*delta
        flag=1;
        noOfcustomers=noOfcustomers+1;
        queueLength=queueLength+1;
        customerId(queueLength)=noOfcustomers;
        customerQueueTimeStamp(customerId(1))=currentSlot;
    else
        flag=0;
    end

    % Check if any teller is free or not, if free then assign the queue
    % head to that teller
    if ~isempty(customerId)
        if tellerStatus(1)==0 || tellerStatus(2)==0 || tellerStatus(3)==0
            temp=find(tellerStatus==0);
            tellerStatus(temp(1))=1;
```

```
            activeBuffer(temp(1))=customerId(1);
            customerArrivalTimeStamp(customerId(1))=currentSlot;
            customerId=customerId(2:end);
            queueLength=queueLength-1;
            flag = 1;
        else
            flag = 0;
        end
    end

    % Check if customer has simple or complex transaction, if simple then
    % calculate service time else calculate service time for complex
    if tellerStatus(1)==1 || tellerStatus(2)==1 || tellerStatus(3)==1
        temp=find(tellerStatus==1);
        for i=1:length(temp)
            if flag==1 && oldBuffer(temp(i))~=activeBuffer(temp(i))
                R=rand(1);
                if R < 0.75
                    n=2;
                    temp1=rand(n,1); % Generate marix of uniform RVs between
0 and 1
                    X=temp1(1,:);
                    for k=2:n
                        X=X.*temp1(k,:); % Multiply samples of uniform RVs
                    end
                    serviceTime(activeBuffer(temp(i)))=ceil(-
log(X)/muSimple);

departureTime(activeBuffer(temp(i)))=customerArrivalTimeStamp(activeBuffer(
temp(i)))+serviceTime(activeBuffer(temp(i)));

                else
                    n=5;
                    temp1=rand(n,1); % Generate marix of uniform RVs between
0 and 1
                    X=temp1(1,:);
                    for k=2:n
                        X=X.*temp1(k,:); % Multiply samples of uniform RVs
                    end
                    serviceTime(activeBuffer(temp(i)))=ceil(-
log(X)/muComplex);

departureTime(activeBuffer(temp(i)))=customerArrivalTimeStamp(activeBuffer(
temp(i)))+serviceTime(activeBuffer(temp(i)));
                end
            end

            if   serviceTime(activeBuffer(temp(i)))   ==   currentSlot   -
customerArrivalTimeStamp(activeBuffer(temp(i)))
                customerDepartureTimeStamp(activeBuffer(temp(i)))         =
currentSlot;
                customerServed = customerServed + 1;
                activeBuffer(temp(i)) = 0;
                tellerStatus(temp(i)) = 0;
            end

        end
        oldBuffer=activeBuffer;
    end
    queue(currentSlot)=queueLength;
```

```
end
```

Output:

Arrival rate λ = 1/60, Simulation time = 3600 seconds

	Mean	Variance
Queue Length	0	0
Waiting Time	0	0

No of customer arrived = 56
No of customers with simple transactions = 38
No of customers with complex transactions = 18
No of customers served = 56

Arrival rate λ = 1/10, Simulation time = 3600 seconds

	Mean	Variance
Queue Length	0.0061	0.0090
Waiting Time	0.1657	3.0424

No of customer arrived = 344
No of customers with simple transactions = 263
No of customers with complex transactions = 81
No of customers served = 342

Arrival rate λ = 1/5, Simulation time = 3600 seconds

	Mean	Variance
Queue Length	0.1171	0.1983
Waiting Time	292.211	52084.6

No of customer arrived = 704
No of customers with simple transactions = 535
No of customers with complex transactions = 169
No of customers served = 703

Arrival rate λ = 1/2, Simulation time = 3600 seconds

	Mean	Variance
Queue Length	23.6845	147.99
Waiting Time	8438.06	135648.6

No of customer arrived = 1780
No of customers with simple transactions = 1359
No of customers with complex transactions = 410
No of customers served = 1766

Part B

Code:

```
clc
clear all
close all
lambda=1/60; % arrival rate per second (1 minute per packet)
T=1*3600; % simulation time in second (10 hours)
delta=0.1; % simulation step size in second
simulationTime=T/delta; % number of simulation steps
currentSlot=0; % Simulation time
noOfcustomers=0; % Total number of customers
noOfSimpleCustomers=0;
noOfComplexCustomers=0;
tellerStatus=zeros(3,1); % Tells whether teller is active or not
activeBuffer=zeros(3,1); % Tells which customer is currently being served
oldBuffer=zeros(3,1);
customerId1=[]; % Stores the customer IDs for queue 1
customerId2=[]; % Stores the customer IDs for queue 2
customerId3=[]; % Stores the customer IDs for queue 3
customerServed=0; % Number of customer served
queueLength=zeros(3,1); % Tells the queue length at each time slot

muSimple=2/60; % mean for simple transactions
muComplex=6/60; % mean for complex transactions

while currentSlot < simulationTime
    currentSlot=currentSlot+1;

    % Check if new customer is arriving
    R=rand(1);
    if R<lambda*delta
        flag=1;
        noOfcustomers=noOfcustomers+1;
        if sum(queueLength==0)
            queueLength(1)=queueLength(1)+1;
            customerId1(queueLength(1))=noOfcustomers;
            customerQueueTimeStamp(customerId1(1))=currentSlot;
        else
            temp=find(queueLength==min(queueLength))
            queueLength(temp)=queueLength(temp)+1;
            if temp==1
                customerId1(queueLength(temp))=noOfcustomers;
                customerQueueTimeStamp(customerId1(1))=currentSlot;
            end
            if temp==2
                customerId2(queueLength(temp))=noOfcustomers;
                customerQueueTimeStamp(customerId2(1))=currentSlot;
            end
            if temp==3
                customerId3(queueLength(temp))=noOfcustomers;
                customerQueueTimeStamp(customerId3(1))=currentSlot;
            end
        end
    end

    % Check if any teller is free or not, if free then assign the queue
    % head to that teller
    if ~isempty(customerId1)
        if tellerStatus(1)==0
```

```
            tellerStatus(1)=1;
            activeBuffer(1)=customerId1(1);
            customerArrivalTimeStamp(customerId1(1))=currentSlot;
            customerId1=customerId1(2:end);
            queueLength(1)=queueLength(1)-1;
            flag = 1;
        else
            flag = 0;
        end
    end
    if ~isempty(customerId2)
        if tellerStatus(2)==0
            tellerStatus(2)=1;
            activeBuffer(2)=customerId2(1);
            customerArrivalTimeStamp(customerId2(1))=currentSlot;
            customerId2=customerId2(2:end);
            queueLength(2)=queueLength(2)-1;
            flag = 1;
        else
            flag = 0;
        end
    end

    if ~isempty(customerId3)
        if tellerStatus(3)==0
            tellerStatus(3)=1;
            activeBuffer(3)=customerId3(1);
            customerArrivalTimeStamp(customerId3(1))=currentSlot;
            customerId3=customerId3(2:end);
            queueLength(3)=queueLength(3)-1;
            flag = 1;
        else
            flag = 0;
        end
    end

    % Check if customer has simple or complex transaction, if simple then
    % calculate service time else calculate service time for complex
    %if sum(tellerStatus>0)
    if tellerStatus(1)==1 || tellerStatus(2)==1 || tellerStatus(3)==1
        temp=find(tellerStatus==1);
        for i=1:length(temp)
            if flag==1 && oldBuffer(temp(i))~=activeBuffer(temp(i))
                R=rand(1);
                if R < 0.75
                    noOfSimpleCustomers=noOfSimpleCustomers+1;
                    n=2;
                    temp1=rand(n,1); % Generate marix of uniform RVs between
0 and 1
                    X=temp1(1,:);
                    for k=2:n
                        X=X.*temp1(k,:); % Multiply samples of uniform RVs
                    end
                    serviceTime(activeBuffer(temp(i)))=ceil(-
log(X)/muSimple);

departureTime(activeBuffer(temp(i)))=customerArrivalTimeStamp(activeBuffer(
temp(i)))+serviceTime(activeBuffer(temp(i)));

                else
                    noOfComplexCustomers=noOfComplexCustomers+1;
```

```
                    n=5;
                    temp1=rand(n,1); % Generate marix of uniform RVs between
0 and 1
                    X=temp1(1,:);
                    for k=2:n
                        X=X.*temp1(k,:); % Multiply samples of uniform RVs
                    end
                    serviceTime(activeBuffer(temp(i)))=ceil(-
log(X)/muComplex);

departureTime(activeBuffer(temp(i)))=customerArrivalTimeStamp(activeBuffer(
temp(i)))+serviceTime(activeBuffer(temp(i)));
                end
            end

            if   serviceTime(activeBuffer(temp(i)))   ==   currentSlot   -
customerArrivalTimeStamp(activeBuffer(temp(i)))
                customerDepartureTimeStamp(activeBuffer(temp(i)))         =
currentSlot;
                customerServed = customerServed + 1;
                activeBuffer(temp(i)) = 0;
                tellerStatus(temp(i)) = 0;
            end

        end
        oldBuffer=activeBuffer;
    end
        queue(currentSlot)=sum(queueLength);
end
```

Output:

Arrival rate λ = 1/60, Simulation time = 3600 seconds

	Mean	Variance
Queue Length	0.0040	0.0040
Waiting Time	4.42	359.5

No of customer arrived = 55
No of customers with simple transactions = 42
No of customers with complex transactions = 13
No of customers served = 55

Arrival rate λ = 1/10, Simulation time = 3600 seconds

	Mean	Variance
Queue Length	0.1947	0.1801
Waiting Time	19.3591	1210.1

No of customer arrived = 390
No of customers with simple transactions = 302
No of customers with complex transactions = 88
No of customers served = 390

Arrival rate λ = 1/5, Simulation time = 3600 seconds

	Mean	Variance
Queue Length	0.4682	0.4182

Waiting Time	113.82	1851600

No of customer arrived = 720
No of customers with simple transactions = 535
No of customers with complex transactions = 185
No of customers served = 719

Arrival rate λ = 1/2, Simulation time = 3600 seconds

	Mean	Variance
Queue Length	18.9409	156.61
Waiting Time	5200.7	104100000

No of customer arrived = 1803
No of customers with simple transactions = 1326
No of customers with complex transactions = 460
No of customers served = 1783

Part C

Code:

```
clc
clear all
close all
lambda=1/60; % arrival rate per second (1 minute per packet)
T=1*3600; % simulation time in second (10 hours)
delta=0.1; % simulation step size in second
simulationTime=T/delta; % number of simulation steps
currentSlot=0; % Simulation time
noOfcustomers=0; % Total number of customers
noOfSimpleCustomers=0;
noOfComplexCustomers=0;
tellerStatus=zeros(3,1); % Tells whether teller is active or not
activeBuffer=zeros(3,1); % Tells which customer is currently being served
transactionType=zeros(3,1); % Tells the transaction type of customer
oldBuffer=zeros(3,1);
customerId1=[]; % Stores the customer IDs for simple transactions
customerId2=[]; % Stores the customer IDs for complex transactions
customerServed=0; % Number of customer served
queueLength=zeros(2,1); % Tells the queue length at each time slot

muSimple=2/60; % mean for simple transactions
muComplex=6/60; % mean for complex transactions

while currentSlot < simulationTime
    currentSlot=currentSlot+1;

    % Check if new customer is arriving
    R=rand(1);
    if R<lambda*delta
        flag=1;
        noOfcustomers=noOfcustomers+1;
        R=rand(1);
        if R < 0.75
            noOfSimpleCustomers=noOfSimpleCustomers+1;
            queueLength(1)=queueLength(1)+1;
```

```
            customerId1(queueLength(1))=noOfcustomers;
            customerQueueTimeStamp(customerId1(1))=currentSlot;
        else
            noOfComplexCustomers=noOfComplexCustomers+1;
            queueLength(2)=queueLength(2)+1;
            customerId2(queueLength(2))=noOfcustomers;
            customerQueueTimeStamp(customerId2(1))=currentSlot;
        end
    end

    % Check if any teller is free or not, if free then assign the queue
    % head to that teller
    if length(customerId1)==1
        if tellerStatus(1)==0|| tellerStatus(2)==0 || tellerStatus(3)==0
            temp=find(tellerStatus==0);
            tellerStatus(temp(1))=1;
            activeBuffer(temp(1))=customerId1(1);
            transactionType(temp(1))=1;
            customerArrivalTimeStamp(customerId1(1))=currentSlot;
            customerId1=customerId1(2:end);
            queueLength(1)=queueLength(1)-1;
            flag = 1;

        else
            flag=0;
        end
    end

    if length(customerId2)==1
        if tellerStatus(2)==0 || tellerStatus(3)==0
            temp=find(tellerStatus(2:3)==0);
            tellerStatus(temp(1))=1;
            activeBuffer(temp(1))=customerId2(1);
            transactionType(temp(1))=2;
            customerArrivalTimeStamp(customerId2(1))=currentSlot;
            customerId2=customerId2(2:end);
            queueLength(2)=queueLength(2)-1;
            flag = 1;
        else
            flag=0;
        end
    end

    if ~isempty(customerId1) && ~isempty(customerId2)
        if tellerStatus(2)==0 || tellerStatus(3)==0
            temp=find(tellerStatus(2:3)==0);
            R=rand(1);
            if R<0.5
                tellerStatus(temp(1))==1;
                activeBuffer(temp(1))=customerId1(1);
                transactionType(temp(1))=1;
                customerArrivalTimeStamp(customerId1(1))=currentSlot;
                customerId1=customerId1(2:end);
                queueLength(1)=queueLength(1)-1;
                flag = 1;
            else
                tellerStatus(temp(1))==1;
                activeBuffer(temp(1))=customerId2(1);
                transactionType(temp(1))=2;
                customerArrivalTimeStamp(customerId2(1))=currentSlot;
```

```
                customerId2=customerId2(2:end);
                queueLength(2)=queueLength(2)-1;
                flag = 1;
            end
        else
            flag = 0;
        end
    end

    if (~isempty(customerId1) && isempty(customerId2)) ||...
        (isempty(customerId1) && ~isempty(customerId2))
        if ~isempty(customerId1)
            if tellerStatus(2)==0 || tellerStatus(3)==0
                temp=find(tellerStatus(2:3)==0);
                tellerStatus(temp(1))=1;
                activeBuffer(temp(1))=customerId1(1);
                transactionType(temp(1))=1;
                customerArrivalTimeStamp(customerId1(1))=currentSlot;
                customerId1=customerId1(2:end);
                queueLength(1)=queueLength(1)-1;
                flag = 1;
            else
                flag = 0;
            end
        else
            if tellerStatus(2)==0 || tellerStatus(3)==0
                temp=find(tellerStatus(2:3)==0);
                tellerStatus(temp(1))==1;
                activeBuffer(temp(1))=customerId2(1);
                transactionType(temp(1))=2;
                customerArrivalTimeStamp(customerId2(1))=currentSlot;
                customerId2=customerId2(2:end);
                queueLength(2)=queueLength(2)-1;
                flag = 1;
            else
                flag = 0;
            end
        end
    end


    % Check if customer has simple or complex transaction, if simple then
    % calculate service time else calculate service time for complex
    %if sum(tellerStatus>0)
    if tellerStatus(1)==1 || tellerStatus(2)==1 || tellerStatus(3)==1
        temp=find(tellerStatus==1);
        for i=1:length(temp)
            if flag==1 && oldBuffer(temp(i))~=activeBuffer(temp(i))
                if transactionType(temp(i))==1;
                    n=2;
                    temp1=rand(n,1); % Generate marix of uniform RVs between
0 and 1
                    X=temp1(1,:);
                    for k=2:n
                        X=X.*temp1(k,:); % Multiply samples of uniform RVs
                    end
                    serviceTime(activeBuffer(temp(i)))=ceil(-
log(X)/muSimple);

departureTime(activeBuffer(temp(i)))=customerArrivalTimeStamp(activeBuffer(
temp(i)))+serviceTime(activeBuffer(temp(i)));
```

```
                else
                    n=5;
                    temp1=rand(n,1); % Generate marix of uniform RVs between
0 and 1
                    X=temp1(1,:);
                    for k=2:n
                        X=X.*temp1(k,:); % Multiply samples of uniform RVs
                    end
                    serviceTime(activeBuffer(temp(i)))=ceil(-
log(X)/muComplex);

departureTime(activeBuffer(temp(i)))=customerArrivalTimeStamp(activeBuffer(
temp(i)))+serviceTime(activeBuffer(temp(i)));
                end
            end

            if   serviceTime(activeBuffer(temp(i)))   ==   currentSlot   -
customerArrivalTimeStamp(activeBuffer(temp(i)))
                customerDepartureTimeStamp(activeBuffer(temp(i)))         =
currentSlot;
                customerServed = customerServed + 1;
                activeBuffer(temp(i)) = 0;
                tellerStatus(temp(i)) = 0;
            end

        end
        oldBuffer=activeBuffer;
    end
        queue(currentSlot)=sum(queueLength);
end
```

Output:

Arrival rate λ = 1/60, Simulation time = 3600 seconds

	Mean	Variance
Queue Length	0	0
Waiting Time	0	0

No of customer arrived = 62
No of customers with simple transactions = 48
No of customers with complex transactions = 14
No of customers served = 62

Arrival rate λ = 1/10, Simulation time = 3600 seconds

	Mean	Variance
Queue Length	0.0015	0.0015
Waiting Time	0.1507	3.98

No of customer arrived = 365
No of customers with simple transactions = 274
No of customers with complex transactions = 91
No of customers served = 364

Arrival rate λ = 1/5, Simulation time = 3600 seconds

	Mean	Variance
Queue Length	0.0359	0.0674
Waiting Time	94.0375	1131900

No of customer arrived = 694
No of customers with simple transactions = 533
No of customers with complex transactions = 161
No of customers served = 693

Arrival rate λ = 1/2, Simulation time = 3600 seconds

	Mean	Variance
Queue Length	0.6248	1.6083
Waiting Time	3068.3	6300100

No of customer arrived = 1819
No of customers with simple transactions = 1359
No of customers with complex transactions = 460
No of customers served = 1817

Conclusion

By comparing all the results of parts A, B and C, we can see that the waiting time for the strategy C is minimum. Also, the mean of queue length is less for strategy C compared to others. The total number of customers arrived depends on the arrival rate λ. If the arrival rate is small then the mean and variance of waiting time and queue length are smaller for strategy.