1) How much time did you spend to do:

- Analyze the problem, determine specifications, and create your design

Answer - Analysing the problem, determining the specifications and the flow of the lab, was a continuous process. It took me about 2 days. For finally, building my code and then along with finding and fixing errors for the supermarket simulation, took me around 3day. The statistics part of the lab was wrapped up in a little around a 1 day.

2)Design/your solution for each requirement :

Solution for each of the requirements and the logic of my code:

- For generating the number of customers during peak hours and in normal hours and the number of items I have created a function with the help of rand() function to return the number of customers and the number of items.
- The service time(or checkout) is created using the formula: **round(((item*5)/60)+1/6+1.5).**
- The class definition is in the Customer.cc files and the declaration in the Customer.h file.
- Changed the Makefile in thread section. Added the following in the Makefile - Customer.h, Customer.cc and Customer.o
- There are 10 Cashier Lines.
- Cashier 1 and Cashier 2 lines are always open.
- A waiting queue is used to keep a track of all the customers entering the supermarket and as soon as the waiting queue reaches 10, a new line is opened or the customer is sent to the cashier line.
- Customer Exiting list is used to keep track of all the statistics.
- CreateCustomer() is the core of the supermarket simulation(All the major functions are in CreateCustomer function)
- The new customers that are coming into the supermarket are first being appended in the waiting queue. Cashier 1 and Cashier 2 are always open. If there are 10 customers in the waiting queue, a new line is opened and the customer at the front of the waiting queue is removed and appended to the new line. The customer only moves from the waiting queue to the Cashier List if there are 10 customers in waiting queue and moves the customer to the open cashier.
- The logic behind the time at which the customer is removed from the Cashier Line and appended to the Customer Exiting Queue is as follows:

  -The time at which the customer enters the Cashier's Line we start decrementing the service time.

  - When the service time is 0, we remove the customer from the Cashier's Line

  -Append the customer to the Customer Exiting the supermarket queue.

  -Below is the code of the implementation of the same

Code: Implementing when the front customer should be removed from the Cashier Line to the Customer Exiting from supermarket list.

time_toDequeue function is in the Customer's class file.

```
time_toDequeue()
{
        if(checkout>0)
        {
                checkout=checkout-1;
        }
        else
        {
      checkout=0;
        }
}
```

CreateCustomer() is in the threadtest.cc

```
void CreateCustomer(int time){
….
….cashier_checkingout(Cashier1);
    cashier_checkingout(Cashier2);
    cashier_checkingout(Cashier3);
    cashier_checkingout(Cashier4);
    cashier_checkingout(Cashier5);
    cashier_checkingout(Cashier6);
    cashier_checkingout(Cashier7);
    cashier_checkingout(Cashier8);
    cashier_checkingout(Cashier9);
    cashier_checkingout(Cashier10);
}
void cashier_checkingout(List<Customer*> *Cashierlol ){
  if(!Cashierlol->IsEmpty()){
      if(Cashierlol->Front()->get_checkout() != 0)
      {
          Cashierlol->Front()->time_toDequeue();
      }
      else if(Cashierlol -> Front() -> get_checkout() == 0)
      {
          CustomersExiting -> Append(Cashierlol -> RemoveFront());
      }
  }
```

Code : WaitingList to Cashier(Every time a new Customer is entering the supermarket)

If there is any space available in open lines and there are 10 customers in waiting queue then append the customer to the Cashier Line.

This code is in the CreateCustomer()

```
 if(Cashier1->NumInList() <=4)
      {
         int c=cashier_number(Cashier1);
         for(int i=0;i<c;i++){

            if(!WaitingQueue -> IsEmpty()){

               waitingqueuetoCashier(Cashier1);
         }
         }
      }

      else if(Cashier2->NumInList() <=4)
      {
         int c=cashier_number(Cashier2);
         for(int i=0;i<c;i++){
            if(!WaitingQueue -> IsEmpty()){
               waitingqueuetoCashier(Cashier2);
         }
      }
      }

      if(WaitingQueue->NumInList()<=4){

         if(WaitingQueue->NumInList() >= 10){

            int c=cashier_number(Cashier3);
            for(int i=0;i<c;i++){
               waitingqueuetoCashier(Cashier3);
          }
          …… Till Cashier10
      }
//Function from waiting queue to Cashier Queue
void waitingqueuetoCashier(List<Customer*> *Cashierlo){
```

```
   Cashierlo -> Append(WaitingQueue->RemoveFront());
}
```

Code : Hourly Simulation

```
if(i%60==0 && i!=0)
     {
       int a=number_of_open_lines();

cout<<"#################################################################################"<<endl;

          cout<<"For Every Hour "<<endl;


cout<<"#################################################################################"<<endl;

         HourlyStats(i);

          cout<<"Maximum waiting time"<< maxtime_queue<<endl;
          //maxtime_queue=10000;
          cout<<"Minimum waiting time"<< mintime_queue<<endl;
          //mintime_queue=0;
          cout<<"Average customers in waiting queue"<< int(people_waiting_queue/i)<<endl;
          cout<<"Largest customers in waiting queue"<< max_cust<<endl;
          max_people_waiting=0;

          cout<<"Smallest customers in waiting queue"<< min_cust<<endl;
          serviceTimehourly(i);
          tset=0;
          shortestServiceTime();
          minimum_servicetime=100000;

          largestServiceTime();
          maximum_servicetime=0;

         cout<< "Average number of open lines"<<int(a/i)<<endl;
         cout<< "Maximum number of open lines"<<maximum_open_line+2<<endl;
          maximum_open_line=0;    }
```

Code : Entire Summary Simulation

```
if(i==300){
        cout<<endl;

cout<<"################################################################################
###"<<endl;
        cout<<"Entire Simulation"<<endl;

cout<<"################################################################################
###"<<endl;



        cout<< "Average Waiting Time is" << int(average_waiting_time/i)<<endl;
        cout<< "Maximum Number of People in Waiting Queue" << max_entire_waiting <<endl;
        serviceTimehourly(i);
        entiresummary_largestServiceTime();
        entiresummary_shortestServiceTime();
        cout<<"Smallest Waiting Time is "<<smallest_waiting_time_for_entire_simulation<<endl;

cout<<"################################################################################
###"<<endl;

    }
```

For every hour,
-Numbers of customers arriving for checkout - I have created an CustomerExiting List, in which after the customer is done with the checkout, customer is removed from the cashier line and appended to CustomerExiting List. The number of customers arriving for checkout would be number of customers done with checkout. It is found out by iterating the CustomerExiting and calculating the number of people at every hour.

Average numbers of customers arriving for checkout
= Number of people in CustomerExiting list(at every hour) /(time-60,120,180,240,300)

(checkout_customer is a global variable)
```
Void HourlyStats(){
….

 cout<< "Average number of customers arriving for checkout" <<
int(checkout_customer/time) << endl;
….
}
```

-The average,smallest and longest waiting time is calculated using the people in the waiting queue only. Eg :-

```
cout<<"Average customers in waitingqueue"<<int(people_waiting_queue/i)<<endl;
```

For the average waiting time, I have calculated the sum of number of people in waiting queue for a particular minute. I have calculated the sum for 60mins. Example if there are 10 people in the waiting queue for the first min and 8 people in waiting queue for the second min then the total average waiting time would be 18/2=9

-The shortest,longest and average service time is the time of the checkout for every customer. The largest and shortest time would be usually 2-5mins as the value of **round(((item*5)/60)+1/6+1.5)** would always range from 2-5.

-The largest,smallest and average number of customers in the waiting queue is found out using the NumInList() in the WaitingQueue. For every hour , the variables used are reset.

-For average number and max of open lines, this function is used and for average time is divided by the total open lines.

Code : Number of Open Lines

```
int number_of_open_lines(){
   if(!Cashier1->IsEmpty()){
      total_number_open_lines=total_number_open_lines+1;
   }
   if(!Cashier2->IsEmpty()){
      total_number_open_lines=total_number_open_lines+1;
   }
   if(!Cashier3->IsEmpty()){
      total_number_open_lines=total_number_open_lines+1;
   }
   if(!Cashier4->IsEmpty()){
      total_number_open_lines=total_number_open_lines+1;
   }
   if(!Cashier5->IsEmpty()){
      total_number_open_lines=total_number_open_lines+1;
   }
   if(!Cashier6->IsEmpty()){
      total_number_open_lines=total_number_open_lines+1;
   }
   if(!Cashier7->IsEmpty()){
      total_number_open_lines=total_number_open_lines+1;
   }
   if(!Cashier8->IsEmpty()){
      total_number_open_lines=total_number_open_lines+1;
   }
   if(!Cashier9->IsEmpty()){
      total_number_open_lines=total_number_open_lines+1;
   }
   if(!Cashier10->IsEmpty()){
      total_number_open_lines=total_number_open_lines+1;
   }
   return total_number_open_lines;
}
```

-For average time each cashier will more than 3 customers standing in line, I used the following function

```
 cout<<"Average time each casher will have more than 3 customers standing in line"<<
int(time_cashier_with_more_than_3/time)<< endl;

//Function checking for time more than 3 customers in the List
int more_than_3()
{
   if(Cashier1->NumInList()>3){
      time_cashier_with_more_than_3=time_cashier_with_more_than_3+1;
   }
   if(Cashier2->NumInList()>3){
      time_cashier_with_more_than_3=time_cashier_with_more_than_3+1;
   }
   if(Cashier3->NumInList()>3){
      time_cashier_with_more_than_3=time_cashier_with_more_than_3+1;
   }
   if(Cashier4->NumInList()>3){
      time_cashier_with_more_than_3=time_cashier_with_more_than_3+1;
   }
   if(Cashier5->NumInList()>3){
      time_cashier_with_more_than_3=time_cashier_with_more_than_3+1;
   }
   if(Cashier6->NumInList()>3){
      time_cashier_with_more_than_3=time_cashier_with_more_than_3+1;
   }
   if(Cashier7->NumInList()>3){
      time_cashier_with_more_than_3=time_cashier_with_more_than_3+1;
   }
   if(Cashier8->NumInList()>3){
      time_cashier_with_more_than_3=time_cashier_with_more_than_3+1;
   }
   if(Cashier9->NumInList()>3){
      time_cashier_with_more_than_3=time_cashier_with_more_than_3+1;
   }
   if(Cashier10->NumInList()>3){
      time_cashier_with_more_than_3=time_cashier_with_more_than_3+1;
   }
   return time_cashier_with_more_than_3;
```

Testing and debug:

For testing and fixing errors, first I created the customers in the first 60 minutes and printed all the list to see if all the functionality is working as per the requirement and my logic. You can uncomment the code and call the function **void print_List(//pass the list here);** if you want to see any list. The result are little absurd in the last hour of the simulation because at 299th minute, there are a lot of people in the waiting list.

Output :
Hour 1:

```
jsavla@lcs-vc-cis486:~/nachos/code/build.linux$ ./nachos -K
################################################################################
For Every Hour
################################################################################

Average number of customers arriving for checkout2
Average Waiting time every hour0
Average time each casher will have more than 3 customers standing in line3
Maximum waiting time10
Minimum waiting time1
Average customers in waiting queue8
Largest customers in waiting queue11
Smallest customers in waiting queue0
Average Service Time3
Shortest Waiting Time Every Hour2
Largest Service Time5
Average number of open lines6
Maximum number of open lines10
################################################################################
```

Hour 2

```
################################################################################
For Every Hour
################################################################################

Average number of customers arriving for checkout3
Average Waiting time every hour0
Average time each casher will have more than 3 customers standing in line3
Maximum waiting time19
Minimum waiting time1
Average customers in waiting queue8
Largest customers in waiting queue10
Smallest customers in waiting queue0
Average Service Time3
Shortest Waiting Time Every Hour2
Largest Service Time5
Average number of open lines7
Maximum number of open lines10
```

Hour 3:

```
################################################################################
For Every Hour
################################################################################

Average number of customers arriving for checkout4
Average Waiting time every hour1
Average time each casher will have more than 3 customers standing in line3
Maximum waiting time313
Minimum waiting time1
Average customers in waiting queue54
Largest customers in waiting queue288
Smallest customers in waiting queue0
Average Service Time3
Shortest Waiting Time Every Hour2
Largest Service Time5
Average number of open lines7
Maximum number of open lines10
```

Hour 4:

```
################################################################################
For Every Hour
################################################################################

Average number of customers arriving for checkout6
Average Waiting time every hour3
Average time each casher will have more than 3 customers standing in line4
Maximum waiting time902
Minimum waiting time1
Average customers in waiting queue151
Largest customers in waiting queue584
Smallest customers in waiting queue0
Average Service Time3
Shortest Waiting Time Every Hour2
Largest Service Time5
Average number of open lines7
Maximum number of open lines10
```

Hour 5 :

```
################################################################################
For Every Hour
################################################################################

Average number of customers arriving for checkout7
Average Waiting time every hour4
Average time each casher will have more than 3 customers standing in line4
Maximum waiting time1491
Minimum waiting time1
Average customers in waiting queue238
Largest customers in waiting queue598
Smallest customers in waiting queue0
Average Service Time3
Shortest Waiting Time Every Hour2
Largest Service Time5
Average number of open lines7
Maximum number of open lines10
```

Entire Summary Simulation :

```
################################################################################
Entire Simulation
################################################################################
Average Waiting Time is4
Maximum Number of People in Waiting Queue598
Average Service Time3

Largest Service Time For Entire Simulation5
Shortest Service Time For Entire Simulation : 2
Smallest Waiting Time is 0
################################################################################
```