1.Objective:

To implement a system which will handle requests from passengers boarding in City A, B, C and D for seats on the flight whose route as follows: City A → City B → City C → City D → City E. City E is the last destination and some passengers will get off in the intermediate cities (B, C and D), and the system will assign these available seats according to the requests. We are assuming the followings:

- The aircraft has **only 20 seats**, which are equally preferable to the passengers (No seat selection)
    - Each seat has unique number (for example, 101 to 120)
- 5 requests at each city A, B, C and D.
    - For each request, the number of passengers should be randomly generated between 1 and 5.
    - Every passenger in the same request has the same itinerary; all passengers in the same party will get off at the same city.
    - The requests are granted in first-come-first-serve basis.
- At each intermediate city, some passenger(s) will get off and their seats will become available.
- If there are not enough seats to satisfy a request, that request will be discarded and next one will be examined until there is neither any seat nor request left.

2.What has been developed:

- Understood the problem statement.
- Created the new files for request class. Header file request.h contains the declaration and request.cc contains the definition.
- All the variables declared as private members and getter, setter, constructor as public members. Added definition in request.cc

Request.h:

```
#include "debug.h"
#include "list.h"
class request
{
	private:
		int uniqueId;
		int passenger;
		char departure;
		char destination;
		List<int>* seatsAssign;
	public:
		~request();
		request(int s, int p, char l, char c, List<int> *m );
```

```
            void setUniqueId(int s);
            void setPassenger(int p);
            void setDest(char c);
            void setDeparture(char l);
            void setSeatsAssign(List<int>* m);

            int getUniqueId();
            int getPassenger();
            char getDest();
            char getDeparture();
            List<int>* getSeatsAssign();
};
```

Request.cc:

```
*/
#include "request.h"

//Costructor for request class
request::request(int s, int p, char l, char c, List<int>* m){
      uniqueId = s;
      passenger = p;
      departure = l;
      destination = c;
      seatsAssign = m;
}

//setter for unique Id
void request::setUniqueId(int s){
      uniqueId = s;
}

//setter for passenger
void request::setPassenger(int p){
      passenger = p;
}

//setter for destination
void request::setDest(char c){
      destination = c;
}

//setter for departure
void request::setDeparture(char l){
      departure = l;
}
```

```
//setter for seatsAssign
void request::setSeatsAssign(List<int>* m){
      seatsAssign = m;
}

//getter for uniqueId
int request::getUniqueId(){
      return uniqueId;
}

//getter for passenger
int request::getPassenger(){
      return passenger;
}

//getter for departure
char request::getDeparture(){
      return departure;
}

//getter for destination
char request::getDest(){
      return destination;
}

//getter for seatsAssign
List<int>* request::getSeatsAssign(){
      return seatsAssign;
}
```

- Added request.h, request.cc and request.o filenames in Makefile.

```
../threads/request.h
../threads/request.cc
THREAD_O = alarm.o kernel.o main.o scheduler.o synch.o thread.o
threadtest.o request.o
```

- Created seats object of Bitmap in threadtest.cc to keep track of the availability of 20 seats.

```
Bitmap *seats = new Bitmap(20);
```

- Created onPlane lists of requests which contain requests Currently on the plane

```
List<request*> *onPlane = new List<request*>();
```

- Created discarded lists of requests which contain Discarded requests because there was no available seat.

```
List<request*> *discarded = new List<request*>();
```

- Created lists for destination cities which stores the reservation threads

```
List<Thread*> *BList = new List<Thread*>();
List<Thread*> *CList = new List<Thread*>();
List<Thread*> *DList = new List<Thread*>();
List<Thread*> *EList = new List<Thread*>();
```

- uId is used for creating request id's

```
int uId = 100;
```

- Created function calculateDest for calculating the destination city according to predefined probabilities.

When Departure city is A: Probability of Destination city as B – 10
Probability of Destination city as C – 20
Probability of Destination city as D – 30
Probability of Destination city as E – 40
When Departure city is B: Probability of Destination city as C – 20
Probability of Destination city as D – 30
Probability of Destination city as E – 50
When Departure city is C: Probability of Destination city as D – 35
Probability of Destination city as D – 65
When Departure city is C: Probability of Destination city as E – 100

```
char calculateDest(char dept){
      switch(dept)
      {
            case 'A': if ( rand() % 100 < 10)
                            return 'B';
                        else if(rand() % 100 < 30)
                            return 'C';
                        else if(rand() % 100 < 60)
                            return 'D';
                        else
                            return 'E';
                      break;
            case 'B': if(rand() % 100 < 20)
                            return 'C';
                        else if(rand() % 100 < 50)
                            return 'D';
                        else
                            return 'E';
                      break;
            case 'C': if(rand() % 100 < 35 )
                            return 'D';
                        else
                            return 'E';
                      break;
            case 'D': return 'E';
                      break;
```

```
            default: return 'E';
        }
}
```

- Created Flight thread which is responsible for simulating the flight and creating Reservation threads. Using single thread for all the flights. Through fork method called SimpleThread function which creates the reservation threads.

```
void
ThreadTest()
{
    Thread *flightA = new Thread("flight");
    flightA->Fork((VoidFunctionPtr) SimpleThread, (void *) 1);
}
```

- In SimpleThread method:
  - srand is used for setting the seed for rand function.
  - Created loop for all the cities and called `callRunThread` method which is used to wake up the threads in list of destination city.
  - Created Reservation thread which is responsible for generating requests, assigning seats to a request, getting a request off the plane. Each Reservation thread deals with one request of each city. Total 5 Reservation threads are generated in for loop for all the cities except 'E'.
  - Through fork method called createRequest which creates the requests.
  - Current thread(Flight thread) is yielded because we are using same thread for all the cities.
  - In the end Discarded request numbers are printed using printDiscard method.

```
void
SimpleThread(int d)
{
      srand(time(0));
      for(char dept = 'A'; dept <= 'E'; dept++){
            callRunThread(dept);

            if(dept != 'E'){
                  for(int i = 1; i<6 ;i++){
                        Thread *res1 = new Thread("reserve");
                  res1->Fork((VoidFunctionPtr) createRequest, (void *)
dept);

            }
            }
      kernel->currentThread->Yield();
      }
      printDiscard();
}
```

- In createRequest method:
  - Incremented the uId for request Id.

- Calculated the number of passengers for the request using the rand function.
- Calculated the destination city using `calculateDest` method.
- Created the list for storing the seat number of passengers of that particular request.
- Using the above parameters created the request.
- Called allocateSeat method to allocate the seats to these passengers.

```
void createRequest(char dept)
{
	printf("\n");
	uId++;
	int passenger = (rand() % 5) + 1;
	char dest = calculateDest(dept);
	List<int> *m = new List<int>();
	request *req1 = new request(uId,passenger,dept,dest,m);
	allocateSeat(req1);
}
```

- In allocateSeat method,
  - Called printReq method to print the request.

```
printf("\nRequest Generated:");
printReq(req);


void printReq(request* req){

	printf("\nReqId: %d,  Passengers: %d, Departure: %c, Destination:
%c ",req->getUniqueId(),req->getPassenger(), req->getDeparture(), req-
>getDest() );
}
```

  - If required seats are available on plane then, set the bits in Bitmap seats and set the seatAssign (list of seat number in request class) for given request.
  - Add the request in onPlane list.
  - Print the seats assigned.
  - Add the current thread(reservation thread) in list of threads of destination city.
  - Set the level af IntOff and put the thread on sleep.
  - clearSeats method is called after the thread is woke up by Scheduler::ReadyToRun.
  - If required seats are not available then, add request in discarded list and finish the current thread.

```
int passenger = req->getPassenger();
char dest = req->getDest();
List<int> *m = new List<int>();
if(passenger <= seats->NumClear()){
	for(int i=0; i<passenger;i++){
		int seat = seats->FindAndSet();
		m->Append(++seat);
	}
	req->setSeatsAssign(m);
```

```
                onPlane->Append(req);
                printSeats(req);
                switch(dest){
                      case'B':BList->Append(kernel->currentThread);
                            break;
                      case 'C':CList->Append(kernel->currentThread);
                            break;
                      case 'D':DList->Append(kernel->currentThread);
                            break;
                      case 'E':EList->Append(kernel->currentThread);
                            break;
                      default:printf("\ndestination : %c", dest);
                }
                kernel->interrupt->SetLevel(IntOff);
                kernel->currentThread->Sleep(FALSE);

                clearSeats(req);
          }else{
                discarded->Append(req);
                printf("\nRequest Discarded: %d", req->getUniqueId() );
                kernel->currentThread->Finish();
          }
    }
```

- Print the assigned seats using printSeats method. This method prints the seats assigned by iterating the list of seats in request. This method also prints the number of seats available and calls printClearBits method to print available seat numbers.

```
void printSeats(request* req){
      ListIterator<int> *iter1 = new ListIterator<int>(req-
>getSeatsAssign());

    printf("\n\nRequest Successfull: %d", req->getUniqueId());
    if(!iter1->IsDone()){
      printf("\nSeat Assigned:");
      for (; !iter1->IsDone(); iter1->Next()) {
       printf(" %d ",iter1->Item()) ;
      }
    }
    printf("\nNumber of Seats available: %d", seats->NumClear());
    printClearBits();
}
```

- Print the available seat numbers using printClearBits method. This method prints the seats available by iterating the Bitmap of seats.

```
void printClearBits(){
      printf("\nAvailable seat numbers:");
```

```
      if(seats->NumClear() > 0){
            for(int i = 0; i <20 ; i++){
                  if(!seats->Test(i))
                        printf(" %d ", i+1);
            }
      }else
            printf(" No Seat available");
}
```

- printOnPlane method prints the request Id's of the request which are on the plane by iterating the onPlane list. This method also prints the Occupancy rate of plane.

```
void printOnPlane(){
      ListIterator<request*> *iter1 = new
ListIterator<request*>(onPlane);
      printf("\nReqId's of Request on the Plane:");
    if(!iter1->IsDone()){
      for (; !iter1->IsDone(); iter1->Next()) {
            request *req = iter1->Item();
            printf(" %d ", req->getUniqueId()) ;
      }
      } else
            printf(" No Passenger on Plane");
      float oRate = (float) (20 - seats->NumClear())*5;
    printf("\nOccupancy Rate : %0.2f %%", oRate);
}
```

- printDiscard method prints the request Id's of the request which are discarded by iterating the discarded list.

```
void printDiscard(){
      ListIterator<request*> *iter1 = new
ListIterator<request*>(discarded);
    if(!iter1->IsDone()){
      printf("\n\nDiscarded ReqId's:");
      for (; !iter1->IsDone(); iter1->Next()) {
            request *req = iter1->Item();
            printf(" %d ", req->getUniqueId()) ;
      }
      }
      printf("\n");
}
```

- callRunThread is called by simpleThread method to woke up the threads in destination city list. This method call runThread method, by passing the list of respective city.

```
void callRunThread(char dept){

      if(dept != 'A'){
            switch(dept){
```

```
                        case 'B':runThread(BList);
                              break;
                        case 'C':runThread(CList);
                              break;
                        case 'D':runThread(DList);
                              break;
                        case 'E':runThread(EList);
                              break;
                        default: printf("\ndept: %c", dept);
                  }
            }
      }
```

- In runThread method, Scheduler::ReadyToRun() is called on all the threads in the list for this city to wake them up. This threads then clear the seats of passenger who are getting off the plane by calling clearSeats method.

```
void runThread(List<Thread*>* list){
      if(!list->IsEmpty()){
            int size = list->NumInList();
            for(int i=0; i < size;i++){
                  Thread *t = list->RemoveFront();
                  kernel->interrupt->SetLevel(IntOff);
                  kernel->scheduler->ReadyToRun(t);
            }
      }
}

clearSeats(req);
```

- clearSeats method clears the bits of the seats for which passengers are getting off and removes the request from onPlane list. After getting passengers off at their destination city, it prints the available seats and id's of request which are on the plane by calling the printClearBits and printOnPlane methods.

```
void clearSeats(request* req){
      onPlane->Remove(req);
      ListIterator<int> *iter1 = new ListIterator<int>(req-
>getSeatsAssign());
      printf("\n\nPassengers of the following request are getting off
at %c:", req->getDest());
    printf("\nReqId: %d", req->getUniqueId());
    if(!iter1->IsDone()){
      printf("   Seats Returned:");
      for (; !iter1->IsDone(); iter1->Next()) {
       printf(" %d ",iter1->Item()) ;
       seats->Clear(iter1->Item() -1);
      }
    }
    printf("\nNumber of Seats available: %d", seats->NumClear());
```

```
        printClearBits();
        printOnPlane();

    }
```

3.How to test your solution:
Run following commands to run the code:

cd nachos/code/build.linux
make
./nachos -K

Output will be displayed on the terminal in the format given in section 5.

Reqeust Id's are generated from 101 to 120.

4.Files modified / Added:

Modified files:
nachos/code/build.linux/Makefile
nachos/code/threads/threadtest.cc

Added files:
nachos/code/threads/request.cc
nachos/code/threads/request.h

5.Output:

Output is printed in following format:

1. **When a request is generated,**
   ReqId, number of passengers,departure city and the destination city is printed. e.g.

   Request Generated:
   ReqId: 101, Passengers: 4, Departure: A, Destination: E

2. **After assigning seats for a request,**
   ReqId, the seat numbers assigned, and total number of available seats and available seat numbers after the assignment are printed. e.g.

   Request Successfull: 101
   Seat Assigned: 1 2 3 4
   Number of Seats available: 16
   Available seat numbers: 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

3. **After request is discarded,**
   ReqId is printed. e.g.

Request Discarded: 108

4. **If it is a destination city for any Reservation thread (request), passengers of the request get off.**
   Printed ReqId of the request, the seat numbers returned, and available numbers after the return

   Passengers of the following request are getting off at C:
   ReqId: 104 Seats Returned: 11 12 13
   Number of Seats available: 4
   Available seat numbers: 11 12 13 20

5. **After getting passengers off at their destination city,**
   Printed ReqId's of all the requests currently on the plane and occupancy rate of seats.

   ReqId's of Request on the Plane: 101 102 105 106 107
   Occupancy Rate : 80.00 %

6. **All the requests discarded throughout the simulation due to lack of available seat.**
   Printed the ReqId's of discarded requests.

   Discarded ReqId's: 108 109 110 112 113 115 119

```
gdamberk@lcs-vc-cis486:~/nachos/code/build.linux$ ./nachos -K


Request Generated:
ReqId: 101,  Passengers: 1, Departure: A, Destination: E

Request Successfull: 101
Seat Assigned: 1
Number of Seats available: 19
Available seat numbers: 2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20

Request Generated:
ReqId: 102,  Passengers: 5, Departure: A, Destination: E

Request Successfull: 102
Seat Assigned: 2  3  4  5  6
Number of Seats available: 14
Available seat numbers: 7  8  9  10  11  12  13  14  15  16  17  18  19  20

Request Generated:
ReqId: 103,  Passengers: 2, Departure: A, Destination: B

Request Successfull: 103
Seat Assigned: 7  8
Number of Seats available: 12
Available seat numbers: 9  10  11  12  13  14  15  16  17  18  19  20

Request Generated:
ReqId: 104,  Passengers: 4, Departure: A, Destination: E

Request Successfull: 104
Seat Assigned: 9  10  11  12
Number of Seats available: 8
Available seat numbers: 13  14  15  16  17  18  19  20

Request Generated:
ReqId: 105,  Passengers: 2, Departure: A, Destination: C

Request Successfull: 105
Seat Assigned: 13  14
Number of Seats available: 6
Available seat numbers: 15  16  17  18  19  20
```

```
Passengers of the following request are getting off at B:
ReqId: 103   Seats Returned: 7  8
Number of Seats available: 8
Available seat numbers: 7  8  15  16  17  18  19  20
ReqId's of Request on the Plane: 101  102  104  105
Occupancy Rate : 60.00 %

Request Generated:
ReqId: 106,  Passengers: 5, Departure: B, Destination: C

Request Successfull: 106
Seat Assigned: 7  8  15  16  17
Number of Seats available: 3
Available seat numbers: 18  19  20

Request Generated:
ReqId: 107,  Passengers: 4, Departure: B, Destination: D
Request Discarded: 107

Request Generated:
ReqId: 108,  Passengers: 1, Departure: B, Destination: D

Request Successfull: 108
Seat Assigned: 18
Number of Seats available: 2
Available seat numbers: 19  20

Request Generated:
ReqId: 109,  Passengers: 3, Departure: B, Destination: E
Request Discarded: 109

Request Generated:
ReqId: 110,  Passengers: 5, Departure: B, Destination: E
Request Discarded: 110
```

```
Passengers of the following request are getting off at C:
ReqId: 105   Seats Returned: 13  14
Number of Seats available: 4
Available seat numbers: 13  14  19  20
ReqId's of Request on the Plane: 101  102  104  106  108
Occupancy Rate : 80.00 %

Passengers of the following request are getting off at C:
ReqId: 106   Seats Returned: 7  8  15  16  17
Number of Seats available: 9
Available seat numbers: 7  8  13  14  15  16  17  19  20
ReqId's of Request on the Plane: 101  102  104  108
Occupancy Rate : 55.00 %

Request Generated:
ReqId: 111,  Passengers: 3, Departure: C, Destination: E

Request Successfull: 111
Seat Assigned: 7  8  13
Number of Seats available: 6
Available seat numbers: 14  15  16  17  19  20

Request Generated:
ReqId: 112,  Passengers: 3, Departure: C, Destination: E

Request Successfull: 112
Seat Assigned: 14  15  16
Number of Seats available: 3
Available seat numbers: 17  19  20

Request Generated:
ReqId: 113,  Passengers: 3, Departure: C, Destination: E

Request Successfull: 113
Seat Assigned: 17  19  20
Number of Seats available: 0
Available seat numbers: No Seat available

Request Generated:
ReqId: 114,  Passengers: 3, Departure: C, Destination: E
Request Discarded: 114
```

```
Request Generated:
ReqId: 115,  Passengers: 3, Departure: C, Destination: E
Request Discarded: 115

Passengers of the following request are getting off at D:
ReqId: 108   Seats Returned: 18
Number of Seats available: 1
Available seat numbers: 18
ReqId's of Request on the Plane: 101  102  104  111  112  113
Occupancy Rate : 95.00 %

Request Generated:
ReqId: 116,  Passengers: 4, Departure: D, Destination: E
Request Discarded: 116

Request Generated:
ReqId: 117,  Passengers: 3, Departure: D, Destination: E
Request Discarded: 117

Request Generated:
ReqId: 118,  Passengers: 1, Departure: D, Destination: E

Request Successfull: 118
Seat Assigned: 18
Number of Seats available: 0
Available seat numbers: No Seat available

Request Generated:
ReqId: 119,  Passengers: 1, Departure: D, Destination: E
Request Discarded: 119

Request Generated:
ReqId: 120,  Passengers: 2, Departure: D, Destination: E
Request Discarded: 120

Passengers of the following request are getting off at E:
ReqId: 101   Seats Returned: 1
Number of Seats available: 1
Available seat numbers: 1
ReqId's of Request on the Plane: 102  104  111  112  113  118
Occupancy Rate : 95.00 %
```

```
Passengers of the following request are getting off at E:
ReqId: 102   Seats Returned: 2  3  4  5  6
Number of Seats available: 6
Available seat numbers: 1  2  3  4  5  6
ReqId's of Request on the Plane: 104  111  112  113  118
Occupancy Rate : 70.00 %

Passengers of the following request are getting off at E:
ReqId: 104   Seats Returned: 9  10  11  12
Number of Seats available: 10
Available seat numbers: 1  2  3  4  5  6  9  10  11  12
ReqId's of Request on the Plane: 111  112  113  118
Occupancy Rate : 50.00 %

Passengers of the following request are getting off at E:
ReqId: 111   Seats Returned: 7  8  13
Number of Seats available: 13
Available seat numbers: 1  2  3  4  5  6  7  8  9  10  11  12  13
ReqId's of Request on the Plane: 112  113  118
Occupancy Rate : 35.00 %

Passengers of the following request are getting off at E:
ReqId: 112   Seats Returned: 14  15  16
Number of Seats available: 16
Available seat numbers: 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16
ReqId's of Request on the Plane: 113  118
Occupancy Rate : 20.00 %

Passengers of the following request are getting off at E:
ReqId: 113   Seats Returned: 17  19  20
Number of Seats available: 19
Available seat numbers: 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  19  20
ReqId's of Request on the Plane: 118
Occupancy Rate : 5.00 %

Passengers of the following request are getting off at E:
ReqId: 118   Seats Returned: 18
Number of Seats available: 20
Available seat numbers: 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20
ReqId's of Request on the Plane: No Passenger on Plane
Occupancy Rate : 0.00 %

Discarded ReqId's: 107  109  110  114  115  116  117  119  120
^C
```

6.Signed disclosure form:

CIS657 Fall 2018

Assignment Disclosure Form

Assignment #: 3

Name: Gauri Amberkar

1. Did you consult with anyone other than instructor or TA/grader on parts of this assignment?

If Yes, please give the details.

- No

2. Did you consult an outside source such as an Internet forum or a book on parts of this assignment?

If Yes, please give the details.

- For rand() function: http://www.cplusplus.com/reference/cstdlib/rand/

- For printing the % sign: https://stackoverflow.com/questions/1860159/how-to-escape-the-percent-sign-in-cs-printf

- For srand(time(0)): https://www.geeksforgeeks.org/rand-and-srand-in-ccpp/

- For loop with char: https://stackoverflow.com/questions/19369415/how-to-i-iterate-a-character-loop-counter-in-java

- For random number between 1-5: https://stackoverflow.com/questions/5891811/generate-random-number-between-1-and-3-in-c

- For random destination with given probability distribution: https://stackoverflow.com/questions/12885356/random-numbers-with-different-probabilities

I assert that, to the best of my knowledge, the information on this sheet is true.

Signature: Gauri Amberkar Date : 10/05/2018