Automated Briyani serving
This code aims to simulate the optimised briyani serving system stipulated by the question
The number of robot chefs, serving tables and students shall be given as input

Assumptions of the code:-

Approach
Entities in this code are Robots chefs, Tables and Students
Each of these entities run on separate threads

Code workings

In main, the thread functions student, serving_table, and chef are called

Two mutex locks are used per serving table which are in arrays chef_table_lock[ ] and table_lock[ ]. Basically the strategy is that each chef, as soon as a briyani portion is prepared tries each of the chef_table_lock and as soon as it finds a lock that is unlocked, it loads one of the r vessels it prepared to the particular serving table. The code snippet for this can be seen below

```
void biryani_ready(int index, int r, int p) {
  for(int i = 0;i<r;){
      int served = 0;
      for(int j = 0;j<N;)
      {
          if(!pthread_mutex_trylock(&chef_table_lock[j]))
          {
              served = 1;
              printf("Vessel %d by chef number %d is served in table %d\n", i+1, index+1,
j+1);
              loaded[j] = p;
              break;
          }
          j+=1;
      }
      i-=1-served;
      i+=1;
  }
}
```

Once a meal has been prepared by the function chef( ), the table_lock is for students. It is initially locked and unlocked only when a table is ready to serve and no other student is having briyani there. Below are the implementations of chef() and ready_to_serve()

```
void *chef(void *index) {
  int *temp=((int*)index);

  int w = random_between(2,5), r = random_between(1,10), p =random_between(2,5);

  printf("Chef %d will now take %d seconds time to prepare %d vessel(s) each with portions to feed for %d student(s)\n", (*temp)+1, w, r, p);
  sleep(w); ///sleep for random time i.e preparation time
  biryani_ready((*temp), r, p);
  return NULL;
}

void ready_to_serve(int index) {
  pthread_mutex_unlock(&table_lock[index]);

  for(;slots[index];){
      ;//spinlock
      ;
  } ///spin lock till slots of index hits 0

  pthread_mutex_lock(&table_lock[index]);
}
```

The lock is unlocked and it is kept unlocked till all the slots are all available, after that its locked again for another round of slots

For serving in the tables, below is the code snippet for it

```
void *serving_table(void *index) {
  int *temp=((int*)index);

  while(1)
  {
      while(!loaded[(*temp)]);

      slots[(*temp)] = min(random_between(1,10), loaded[(*temp)]);
```

```
        printf("Serving table %d has made %d slot(s) available\n", (*temp)+1,
slots[(*temp)]);

        ready_to_serve((*temp));

        loaded[(*temp)] -= slots[(*temp)];

        if(!loaded[(*temp)])
            pthread_mutex_unlock(&chef_table_lock[(*temp)]);

        sleep(1);

    }

}
```

To handle the waiting of student for slot and more, function wait_in_slot() is used who's code snippet is found below. It keeps trying all the locks and the moment it finds one it calls the function student_in_slot() function and then exits the loop as the student has gotten his/her briyani

```
void wait_for_slot(int index) {
  bool served = false;
  printf("Student number %d is waiting for a slot in a table\n", index+1);
  while(!served)
  {
      int i = 0;
      for(;i<N;)
      {
          if(!pthread_mutex_trylock(&table_lock[i]))
          {
              if(slots[i]==0){ ///if slots become 0
                  pthread_mutex_unlock(&table_lock[i]);
                  continue;
              }

              student_in_slot(index, i);
              served = true;
              pthread_mutex_unlock(&table_lock[i]);
```

```
            break;
        }
        i+=1;
    }
  }
}
```

Below is the code implementation of student_in_slot(). The slots are decreased by one because the student has had his briyani

```
void student_in_slot(int index, int table) {
  printf("Student number %d is having briyani at table %d\n", index+1, table+1);
  sleep(2);
  slots[table]-=1;
}
```

Result:-
Below is a sample output produced by the code

Enter number of chefs, serving tables and students
5 3 8
Chef 1 will now take 5 seconds time to prepare 7 vessel(s) each with portions to feed for 4 student(s)
Chef 2 will now take 4 seconds time to prepare 1 vessel(s) each with portions to feed for 3 student(s)
Chef 3 will now take 4 seconds time to prepare 5 vessel(s) each with portions to feed for 3 student(s)
Chef 4 will now take 3 seconds time to prepare 2 vessel(s) each with portions to feed for 2 student(s)
Chef 5 will now take 5 seconds time to prepare 4 vessel(s) each with portions to feed for 4 student(s)
Student number 1 is waiting for a slot in a table
Student number 2 is waiting for a slot in a table
Student number 3 is waiting for a slot in a table
Student number 5 is waiting for a slot in a table
Student number 4 is waiting for a slot in a table
Student number 6 is waiting for a slot in a table
Student number 7 is waiting for a slot in a table
Student number 8 is waiting for a slot in a table

Vessel 1 by chef number 4 is served in table 1
Vessel 2 by chef number 4 is served in table 2
Serving table 2 has made 2 slot(s) available
Student number 8 is having briyani at table 2
Serving table 1 has made 2 slot(s) available
Student number 6 is having briyani at table 1
Vessel 1 by chef number 2 is served in table 3
Serving table 3 has made 4 slot(s) available
Student number 1 is having briyani at table 3
Student number 7 is having briyani at table 2
Student number 2 is having briyani at table 1
Student number 3 is having briyani at table 3
Student number 5 is having briyani at table 3
Serving table 2 has made 2 slot(s) available
Student number 4 is having briyani at table 2
Serving table 1 has made 2 slot(s) available
All students have been fed briyani