ROBOCHEF

[

-> Thread for robochef are created as soon as input is recieved.

-> In each thread chef starts preparing food .

-> When the food is prepared , the chef enters biryani_ready() function and starts seraching for table with empty container.

-> If he finds a table with empty contanier , the chef unloads one of his vessel in container.

```
  while(num>0)
        {
                pthread_mutex_lock(&mutex1);
                for(i=0;i<tablenum;i++)
                {if(table[i]==0)
                        {table[i]+=size;
                                num--;
                                printf("serving conatiner of table %d is refilled by robot chef %d\n",i+1,ind);
                               if(num<=0)
                                       break;
                        }
                }
               pthread_mutex_unlock(&mutex1);

        }
```

->When chef has unloaded  all his vessels he returns from function biryani_ready().

->After returning the chef  starts preparing another batch.

]

TABLE

[

-> When a table thread is cerated it starts checking if its container is filled .

-> When the container of table is filled it generates a random number that denotes number of slots available and enters ready_to_serve() function .

```
    while(z!=1)
            {
                    pthread_mutex_lock(&mutex1);
                    pthread_mutex_lock(&mutex2);
            if(table[ind-1]>0)
            {
                if(table[ind-1]>max)
                {
                    num=max;
                }
                else
                {
                    num=table[ind-1];
                }

                z=1;
                slot=rand()%num+1;
                tabslot[ind-1]+=slot;
            }
            if(z==1)
                printf("serving table %d is ready with %d slot\n",ind,slot);
            pthread_mutex_unlock(&mutex2);
            pthread_mutex_unlock(&mutex1);
        }
```

->It stays in the function until all the slots that were avilable are used.

```
 while(z!=1)
     {
         pthread_mutex_lock(&mutex2);
         if(tabslot[ind-1]==0)
             z=1;
         pthread_mutex_unlock(&mutex2);
     }
```

->After returning , the table starts checking state of its container again.

]

STUDENT

[

-> When a student arrives in mess he/she calls wait_for_slot() funcion.

-> In wait_for_slot() function the student checks if any slot is available at any table.

```
 while(z!=1)
     {
          pthread_mutex_lock(&mutex1);
          pthread_mutex_lock(&mutex2);
         for(i=0;i<tablenum;i++)
```

```
            { if(tabslot[i]>0)
                  {
                        tabslot[i]--;
                        z=1;
                        printf("student %d is assigned a slot on table %d\n",ind,i+1);
                        student_in_slot(ind,i);
                        break;
                  }
            }
          pthread_mutex_unlock(&mutex2);
            pthread_mutex_unlock(&mutex1);

      }
```

-> When the student find the slot he/she calls student_in_slot() to let the table know that he/she has arrived at slot.

->After the student is served the thread is terminated.

]