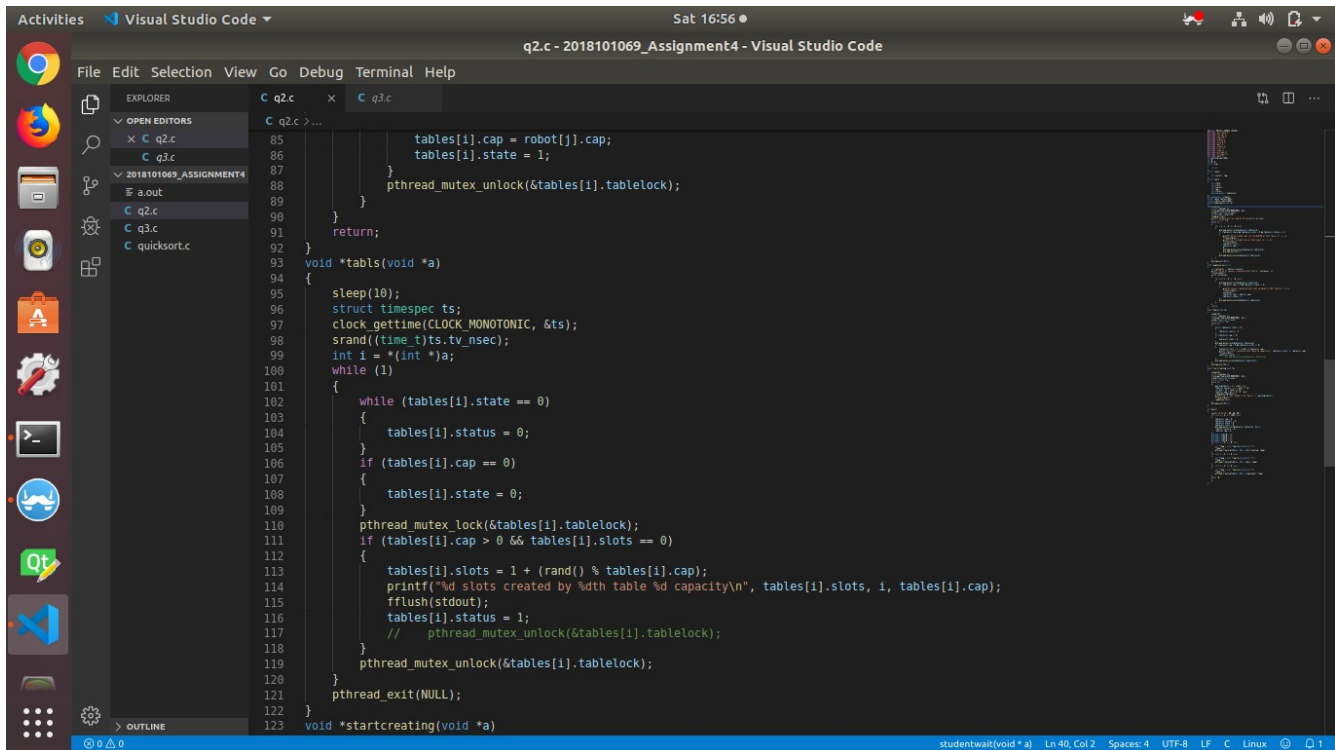


Report for question 2

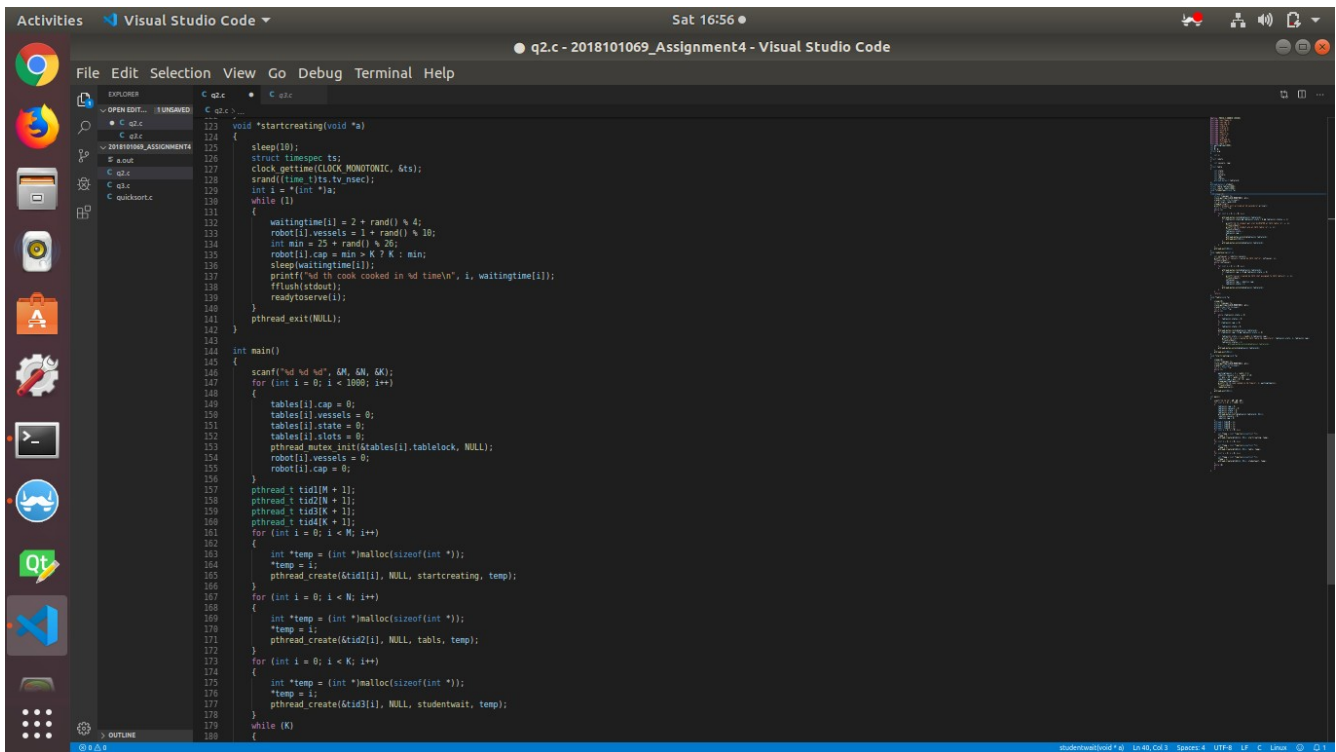


The screenshot shows the Visual Studio Code editor with a C file named `q2.c` open. The code is a multi-threaded program for managing a table with slots. It includes headers for `pthread`, `stdio`, `stdlib`, `unistd`, `time`, and `sys/time`. A `table_t` struct is defined with fields for `cap`, `state`, `status`, and `slots`. The `main` function creates an array of `table_t` structures and calls `startcreating` for each. The `startcreating` function is a `pthread_t` thread that loops while the table's state is 0. Inside the loop, it sets the table's status to 0, checks if the capacity is 0, and if so, sets the state to 0. It then locks the table's mutex, increments the number of slots (up to the capacity), prints the current state, and unlocks the mutex. Finally, it calls `pthread_exit(NULL)`.

```
1  #include <pthread.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <unistd.h>
5  #include <time.h>
6  #include <sys/time.h>
7
8  struct table_t {
9      int cap;
10     int state;
11     int status;
12     int slots;
13 };
14
15 int main() {
16     table_t tables[10];
17     pthread_t threads[10];
18     for (int i = 0; i < 10; i++) {
19         tables[i].cap = 10;
20         tables[i].state = 0;
21         tables[i].status = 0;
22         tables[i].slots = 0;
23         pthread_t thread;
24         pthread_create(&thread, NULL, startcreating, (void *)i);
25         threads[i] = thread;
26     }
27     for (int i = 0; i < 10; i++) {
28         pthread_join(threads[i], NULL);
29     }
30     return 0;
31 }
32
33 void *startcreating(void *a) {
34     int i = (int *)a;
35     while (1) {
36         while (tables[i].state == 0) {
37             tables[i].status = 0;
38             if (tables[i].cap == 0) {
39                 tables[i].state = 0;
40             }
41             pthread_mutex_lock(&tables[i].tablelock);
42             if (tables[i].cap > 0 && tables[i].slots == 0) {
43                 tables[i].slots = 1 + (rand() % tables[i].cap);
44                 printf("Slots created by %dth table %d capacity\n", tables[i].slots, i, tables[i].cap);
45                 fflush(stdout);
46                 tables[i].status = 1;
47                 pthread_mutex_unlock(&tables[i].tablelock);
48             }
49             pthread_mutex_unlock(&tables[i].tablelock);
50         }
51         pthread_exit(NULL);
52     }
53 }
54
55 void *startcreating(void *a)
```

Tables function is function used for threads of tables and creating slots in the table and changing state of table struct

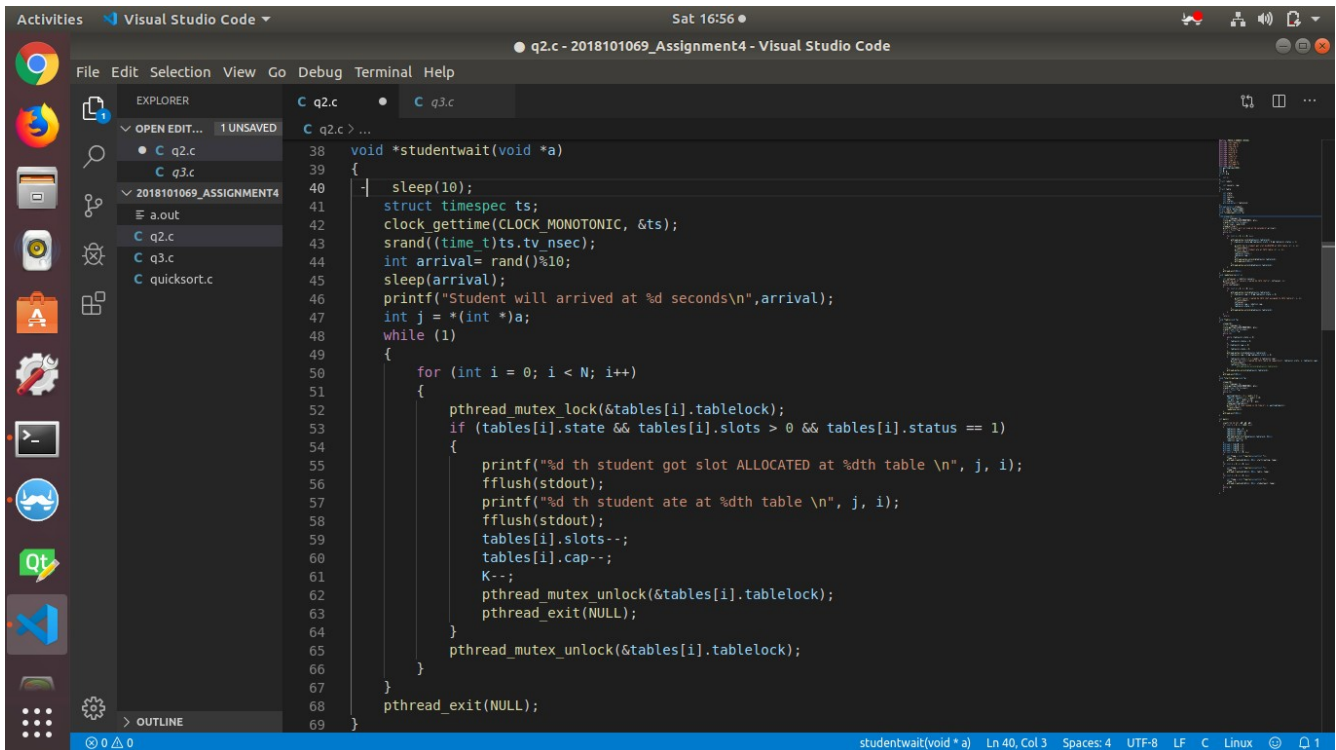
Startcreating function is function used for threads of chefs and is used to call readyto serve function when a chef is finished cooking biryani



The screenshot shows the Visual Studio Code editor with the file `q2.c` open. The code defines a `startcreating` function and a `main` function. The `startcreating` function initializes a robot's waiting time, vessels, and cap, and prints a message. The `main` function scans for input, initializes tables, and creates threads for chefs, students, and tables.

```
121 void *startcreating(void *a)
122 {
123     sleep(10);
124     struct timespec ts;
125     clock_gettime(CLOCK_MONOTONIC, &ts);
126     srand((time_t)ts.tv_nsec);
127     int i = *(int *)a;
128     while (1)
129     {
130         waitingtime[i] = 2 + rand() % 4;
131         robot[i].vessels = 1 + rand() % 10;
132         int min = 25 + rand() % 20;
133         robot[i].cap = min > K ? K : min;
134         sleep(waitingtime[i]);
135         printf("%d th cook cooked in %d time\n", i, waitingtime[i]);
136         fflush(stdout);
137         readytoserve(i);
138     }
139     pthread_exit(NULL);
140 }
141
142 int main()
143 {
144     scanf("%d %d %d", &M, &N, &K);
145     for (int i = 0; i < 1000; i++)
146     {
147         tables[i].cap = 0;
148         tables[i].vessels = 0;
149         tables[i].state = 0;
150         tables[i].slots = 0;
151         pthread_mutex_init(&tables[i].tablelock, NULL);
152         robot[i].vessels = 0;
153         robot[i].cap = 0;
154     }
155     pthread_t tid1[M + 1];
156     pthread_t tid2[N + 1];
157     pthread_t tid3[K + 1];
158     pthread_t tid4[K + 1];
159     for (int i = 0; i < M; i++)
160     {
161         int *temp = (int *)malloc(sizeof(int));
162         *temp = i;
163         pthread_create(&tid1[i], NULL, startcreating, temp);
164     }
165     for (int i = 0; i < N; i++)
166     {
167         int *temp = (int *)malloc(sizeof(int));
168         *temp = i;
169         pthread_create(&tid2[i], NULL, tabs, temp);
170     }
171     for (int i = 0; i < K; i++)
172     {
173         int *temp = (int *)malloc(sizeof(int));
174         *temp = i;
175         pthread_create(&tid3[i], NULL, studentwait, temp);
176     }
177     while (K)
178     {
179     }
180 }
```

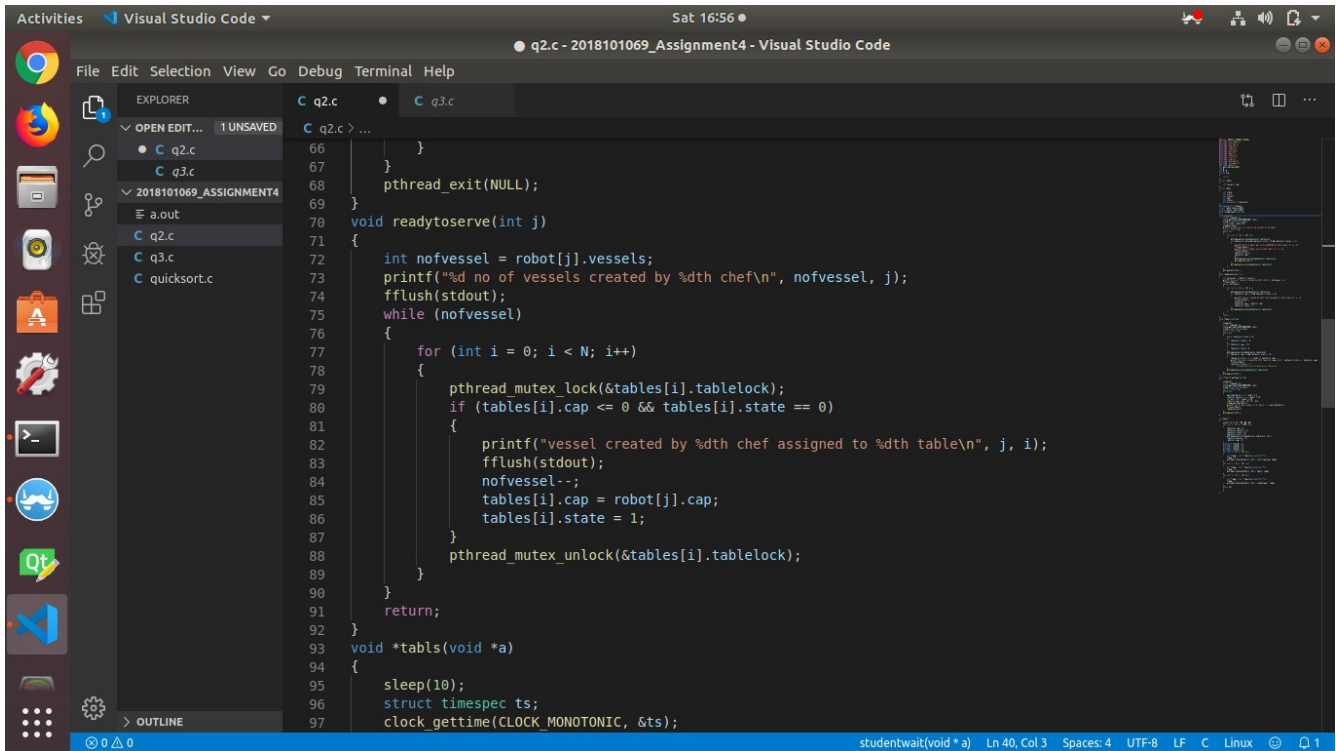
This snippet also shows Main function which is used to create threads for chefs, students and tables. It also uses a while loop to make sure all students gets Biryani.



The screenshot shows the Visual Studio Code editor with the file `q3.c` open. The code defines a `studentwait` function. The function prints a message, checks if a slot is available, and updates the table's state and slots.

```
38 void *studentwait(void *a)
39 {
40     sleep(10);
41     struct timespec ts;
42     clock_gettime(CLOCK_MONOTONIC, &ts);
43     srand((time_t)ts.tv_nsec);
44     int arrival = rand() % 10;
45     sleep(arrival);
46     printf("Student will arrived at %d seconds\n", arrival);
47     int j = *(int *)a;
48     while (1)
49     {
50         for (int i = 0; i < N; i++)
51         {
52             pthread_mutex_lock(&tables[i].tablelock);
53             if (tables[i].state && tables[i].slots > 0 && tables[i].status == 1)
54             {
55                 printf("%d th student got slot ALLOCATED at %dth table \n", j, i);
56                 fflush(stdout);
57                 printf("%d th student ate at %dth table \n", j, i);
58                 fflush(stdout);
59                 tables[i].slots--;
60                 tables[i].cap--;
61                 K--;
62                 pthread_mutex_unlock(&tables[i].tablelock);
63                 pthread_exit(NULL);
64             }
65             pthread_mutex_unlock(&tables[i].tablelock);
66         }
67     }
68     pthread_exit(NULL);
69 }
```

This snippet shows code for threads of student which is used to get students slots and feed students



The screenshot shows the Visual Studio Code interface with a C file named `q2.c` open. The code is written in C and uses pthreads for threading. The function `readytoserve` is defined, which takes an integer `j` as an argument. It calculates the number of vessels created by a chef, prints it, and then enters a loop to create vessels. Inside the loop, it locks a mutex, checks if a table is available, prints a message, decrements the number of vessels, updates the table's capacity and state, and then unlocks the mutex. The function also includes a `sleep(10)` call and a `clock_gettime` call.

```
66     }
67 }
68 pthread_exit(NULL);
69 }
70 void readytoserve(int j)
71 {
72     int nofvessel = robot[j].vessels;
73     printf("%d no of vessels created by %dth chef\n", nofvessel, j);
74     fflush(stdout);
75     while (nofvessel)
76     {
77         for (int i = 0; i < N; i++)
78         {
79             pthread_mutex_lock(&tables[i].tablelock);
80             if (tables[i].cap <= 0 && tables[i].state == 0)
81             {
82                 printf("vessel created by %dth chef assigned to %dth table\n", j, i);
83                 fflush(stdout);
84                 nofvessel--;
85                 tables[i].cap = robot[j].cap;
86                 tables[i].state = 1;
87             }
88             pthread_mutex_unlock(&tables[i].tablelock);
89         }
90     }
91     return;
92 }
93 void *tabs(void *a)
94 {
95     sleep(10);
96     struct timespec ts;
97     clock_gettime(CLOCK_MONOTONIC, &ts);
```

This Snippet is used to assign table to vessel and change state of the TABLE.