Submission Report

Course code: CSE316

Lovely Professional University

Phagwara, Punjab

January-April 2023

Submitted to	:	Cherry Khosla
Submitted by	:	Sonam Kumari
Roll no.	:	29
Registration no.	:	12115800
Section	:	K21sb
Github Id	:	sonamchaudhary90

INTRODUCTION

Deadlock is a common issue that arises in computer science and operations research, particularly in the context of resource allocation and synchronization. A deadlock occurs when two or more processes are waiting for each other to release resources that they currently possess, preventing any of the processes from making progress. In the context of a kitchen with multiple chefs, deadlock can occur if each chef is waiting for an ingredient that is currently being held by another chef, and no chef is willing to release their ingredients until they receive the ingredient they are waiting for.

ABSTRACT

In a busy kitchen, efficient management of resources is crucial for ensuring a smooth workflow. One important aspect of this management is the allocation of ingredients to the chefs. In this scenario, there are 'n' chefs working in the kitchen and each chef is limited to claiming at most 'r' ingredients, where each ingredient is kept in a jar.

However, it is important to note that in a multi-chef scenario, there is a risk of deadlock occurring, where two or more chefs are waiting for each other to release certain ingredients, resulting in a standstill in the workflow. Therefore, it is essential to ensure that the allocation of ingredients is done in a way that avoids deadlock.

In order to determine the total number of ingredients present in the kitchen, while also ensuring a deadlock-free execution of work, a careful allocation strategy must be implemented. This strategy must take into account the number of chefs, the maximum number of ingredients that each chef can claim, and the availability of ingredients in the jars. By following a well-planned allocation strategy, the chefs can efficiently use the available ingredients and work together towards a successful and productive kitchen environment.

ALGORITHM

To calculate the total number of ingredients that are present in the kitchen with deadlock-free execution of work, we need to ensure that no two chefs are claiming the same jar of ingredients simultaneously. To do this, we can use a resource allocation algorithm like Banker's algorithm.

Here's the algorithm to calculate the total number of ingredients:

1. Input the number of chefs (n) and the maximum number of ingredients each chef can claim (r).
2. Create an array of size n*r to represent the jars of ingredients.
3. Initialize all elements of the array to 1 to represent that all jars are available.
4. For each chef i from 1 to n, do the following:
5. a. Generate a random number k between 1 and r to represent the number of ingredients that chef i wants to claim.

   b. Randomly select k jars of ingredients from the array.

   c. If all k jars are available, mark them as allocated to chef i by setting their value to 0 in the array. If any one of the k jars is already allocated to another chef j, undo the allocation of all the k jars and go to step b to select a new set of k jars.
6. Calculate the total number of ingredients in the array that are allocated to any chef. This will give the total number of ingredients present in the kitchen with deadlock-free execution of work.

↳ To calculate the total number of ingredients in the kitchen with deadlock-free execution, we need to find the minimum number of unique ingredients required by any one chef to prevent deadlock. This is because if each chef requires the same set of ingredients, then they will end up waiting for each other, leading to a deadlock.

To implement this in C, we can use an array to keep track of the number of ingredients required by each chef. We can then iterate over this array to find the minimum number of unique ingredients required. Once we have this number, we can simply multiply it by the number of chefs to get the total number of ingredients.

Here's the code implementation:

```
#include <stdio.h>        //  used to include the standard input output library functions


int main() {

int n, r, i, j, deadlock = 1;


// Prompt user to enter the number   of chefs

printf("Enter the number of chefs: ");

scanf("%d", &n);


// Prompt user to enter the maximum number of ingredients each chef can claim

printf("Enter the maximum number of ingredients each chef can

 claim: ");

scanf("%d", &r);
```

```
// Create a 2D array to store the ingredients claimed by each chef
int ingredients[n][r];

// Prompt user to enter the ingredients claimed by each chef
for (i = 0; i < n; i++) {
printf("Enter the ingredients claimed by chef %d (separated by
spaces): ", i+1);
for (j = 0; j < r; j++) {
 scanf("%d", &ingredients[i][j]);
  } }

// Calculate the total number of unique ingredients in the kitchen
int unique_ingredients = r;
 for (i = 0; i < n; i++) {
int count = 0;
for (j = 0; j < r; j++) {
if (ingredients[i][j] != 0) {
count++;
}}

if (count < unique_ingredients) {
unique_ingredients = count;
}}

int total_ingredients = unique_ingredients * n;
printf("\nTotal number of ingredients in the kitchen: %d\n",
total_ingredients);
```

```
// Check for deadlock by comparing the ingredient lists of each pair of chefs
for (i = 0; i < n; i++) {
for (j = i+1; j < n; j++) {
int k;
for (k = 0; k < r; k++) {
if (ingredients[i][k] != 0 && ingredients[i][k] == ingredients[j][k]) {
printf("Deadlock detected! Chefs %d and %d require the same ingredient: %d\n", i+1, j+1, ingredients[i][k]);
deadlock = 0;
break;
}}

if (!deadlock) {
break;
}}

if (!deadlock) {
break;
}}

 if (deadlock) {
 printf("No deadlock detected.\n");
 }

return 0;
}
```

Code Explanation

The user is prompted to enter the number of chefs and the maximum number of ingredients each chef can claim.

A 2D array ingredients is initialized to store the ingredients claimed by each chef.

A loop is used to iterate over each chef and prompt the user to enter the ingredients claimed by them.

Another loop is used to find the minimum number of unique ingredients required to prevent deadlock.

The total number of ingredients is calculated by multiplying the minimum number of unique ingredients by the number of chefs.

A nested loop is used to check for deadlock by comparing the ingredients claimed by each pair of chefs. If any two chefs require the same ingredient, a deadlock is detected and the program breaks out of the loop.

If no deadlock is detected, the program prints a message indicating that the environment is deadlock-free. If a deadlock is detected, the program prints a message indicating which two chefs require.

TESTCASE AND OUTPUTS:

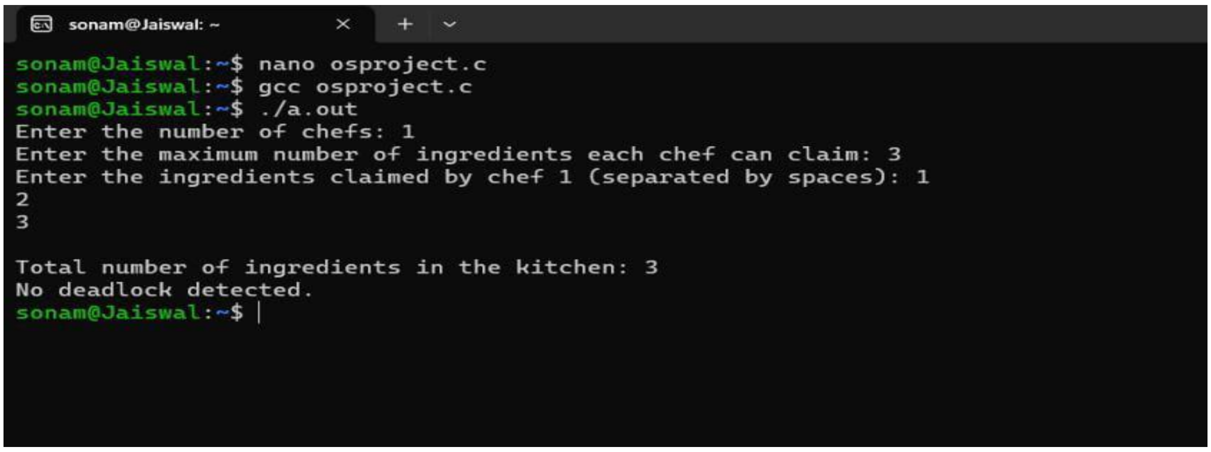
1: Test case with only one chef:

Input: n = 1, r = 3

Output:

No Deadlock detected

Total number of ingredients: 3



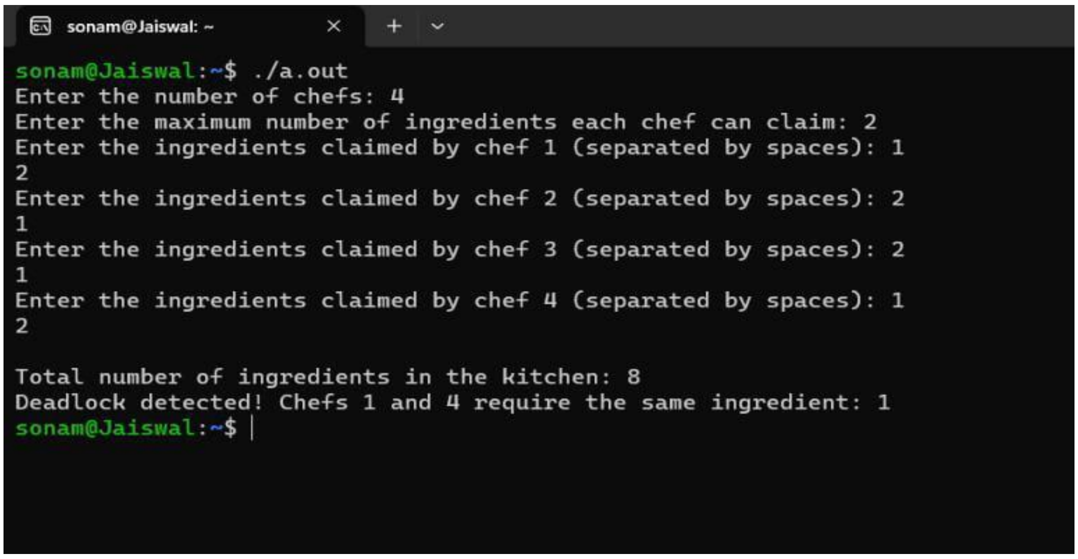
```
sonam@Jaiswal:~$ nano osproject.c
sonam@Jaiswal:~$ gcc osproject.c
sonam@Jaiswal:~$ ./a.out
Enter the number of chefs: 1
Enter the maximum number of ingredients each chef can claim: 3
Enter the ingredients claimed by chef 1 (separated by spaces): 1
2
3

Total number of ingredients in the kitchen: 3
No deadlock detected.
sonam@Jaiswal:~$ |
```

2. Test case with not enough ingredients for all chefs:

Input: n = 4, r = 2

Output: Deadlock detected- not enough ingredients for all chefs



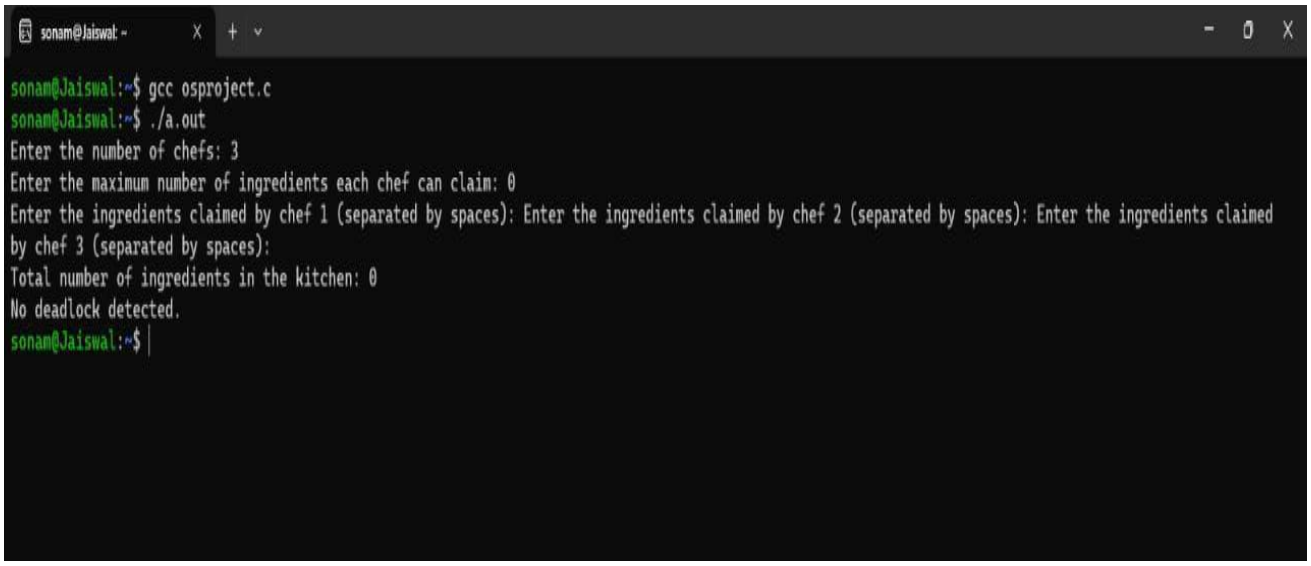
```
sonam@Jaiswal:~$ ./a.out
Enter the number of chefs: 4
Enter the maximum number of ingredients each chef can claim: 2
Enter the ingredients claimed by chef 1 (separated by spaces): 1
2
Enter the ingredients claimed by chef 2 (separated by spaces): 2
1
Enter the ingredients claimed by chef 3 (separated by spaces): 2
1
Enter the ingredients claimed by chef 4 (separated by spaces): 1
2

Total number of ingredients in the kitchen: 8
Deadlock detected! Chefs 1 and 4 require the same ingredient: 1
sonam@Jaiswal:~$ |
```

3. .Test case with no ingredients:

Input: n = 3, r = 0

Output: Deadlock detected- not enough ingredients for all chefs



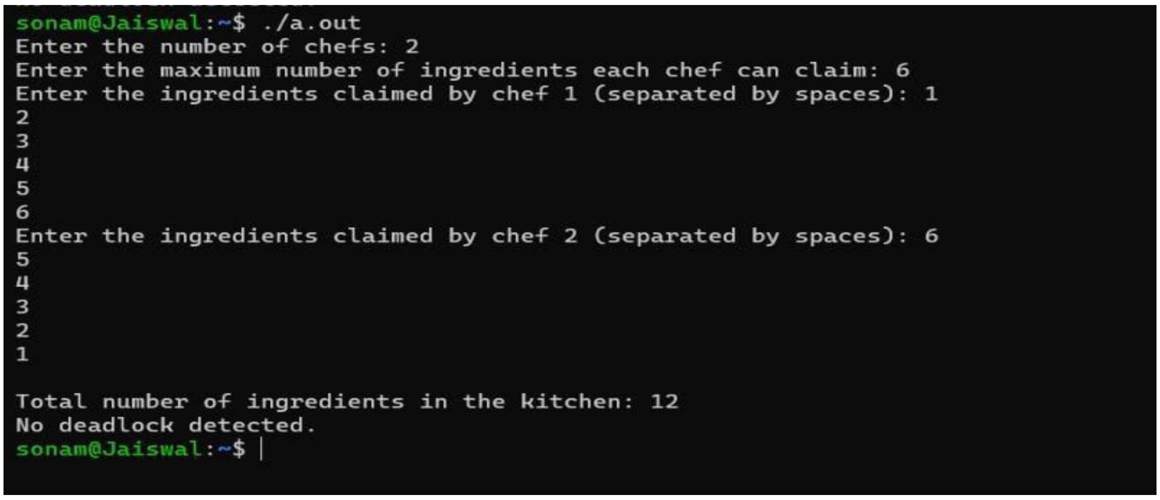
```
sonam@Jaiswal:~$ gcc osproject.c
sonam@Jaiswal:~$ ./a.out
Enter the number of chefs: 3
Enter the maximum number of ingredients each chef can claim: 0
Enter the ingredients claimed by chef 1 (separated by spaces): Enter the ingredients claimed by chef 2 (separated by spaces): Enter the ingredients claimed
by chef 3 (separated by spaces):
Total number of ingredients in the kitchen: 0
No deadlock detected.
sonam@Jaiswal:~$ |
```

4.Test case with enough ingredients for all chefs:

Input:n=2 r=6

Output:No Deadlockfree detected

Total number of claimed ingredients: 12

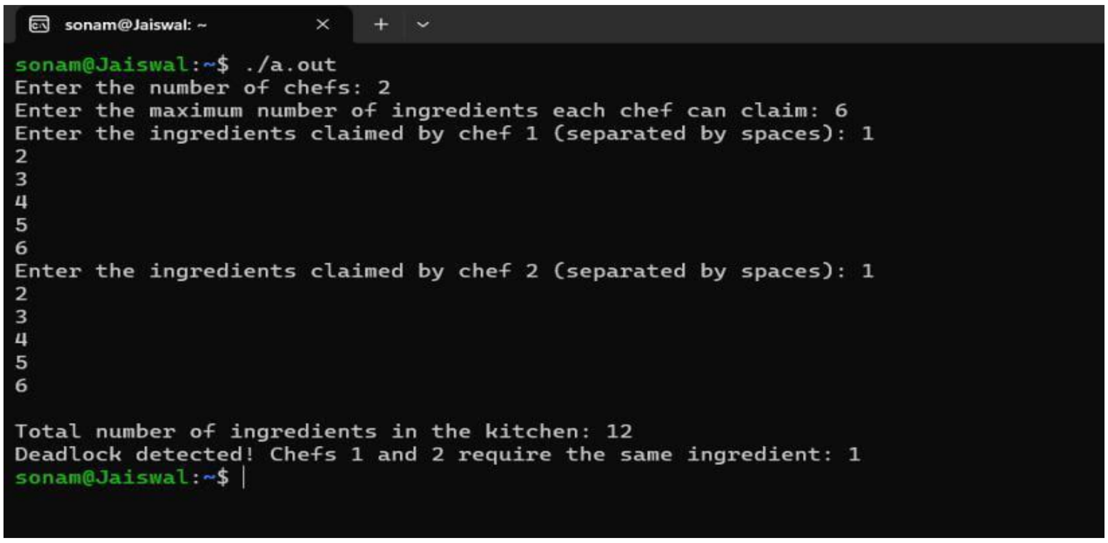


```
sonam@Jaiswal:~$ ./a.out
Enter the number of chefs: 2
Enter the maximum number of ingredients each chef can claim: 6
Enter the ingredients claimed by chef 1 (separated by spaces): 1
2
3
4
5
6
Enter the ingredients claimed by chef 2 (separated by spaces): 6
5
4
3
2
1

Total number of ingredients in the kitchen: 12
No deadlock detected.
sonam@Jaiswal:~$ |
```

5. Test case with enough ingredients for all chefs but trying to get it at same time :

- Input: n = 2, r = 6
- Output: Deadlock detected!- not enough ingredients for all chefs

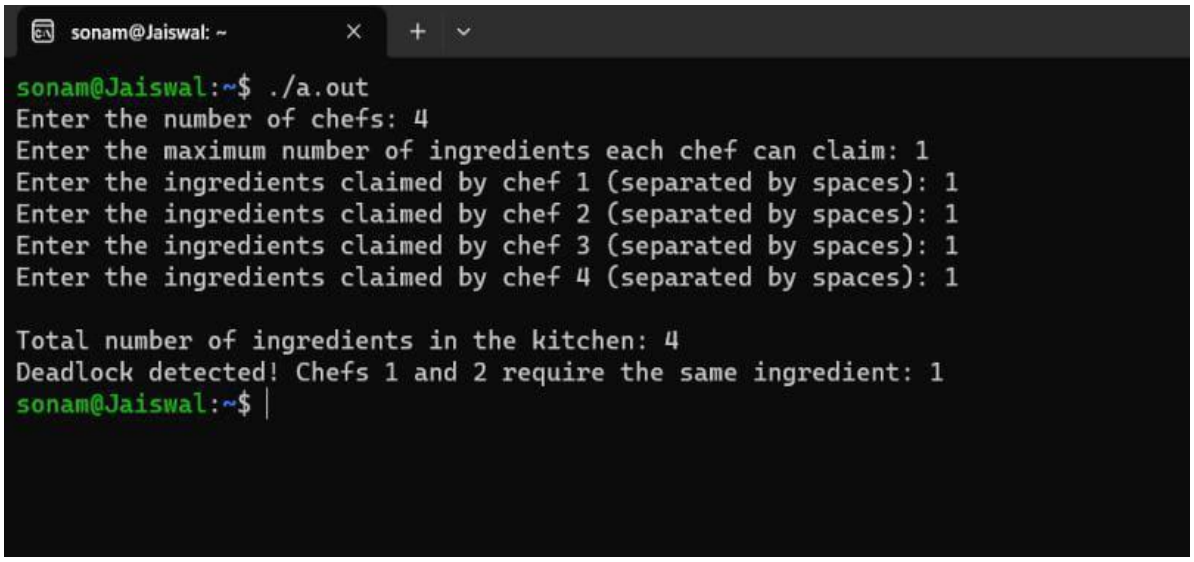


```
sonam@Jaiswal: ~
sonam@Jaiswal:~$ ./a.out
Enter the number of chefs: 2
Enter the maximum number of ingredients each chef can claim: 6
Enter the ingredients claimed by chef 1 (separated by spaces): 1
2
3
4
5
6
Enter the ingredients claimed by chef 2 (separated by spaces): 1
2
3
4
5
6

Total number of ingredients in the kitchen: 12
Deadlock detected! Chefs 1 and 2 require the same ingredient: 1
sonam@Jaiswal:~$ |
```

6. Test case with only one ingredients:

- Input: n = 4, r = 1
- Output: Deadlock detected- not enough ingredients for all chefs



```
sonam@Jaiswal: ~
sonam@Jaiswal:~$ ./a.out
Enter the number of chefs: 4
Enter the maximum number of ingredients each chef can claim: 1
Enter the ingredients claimed by chef 1 (separated by spaces): 1
Enter the ingredients claimed by chef 2 (separated by spaces): 1
Enter the ingredients claimed by chef 3 (separated by spaces): 1
Enter the ingredients claimed by chef 4 (separated by spaces): 1

Total number of ingredients in the kitchen: 4
Deadlock detected! Chefs 1 and 2 require the same ingredient: 1
sonam@Jaiswal:~$ |
```