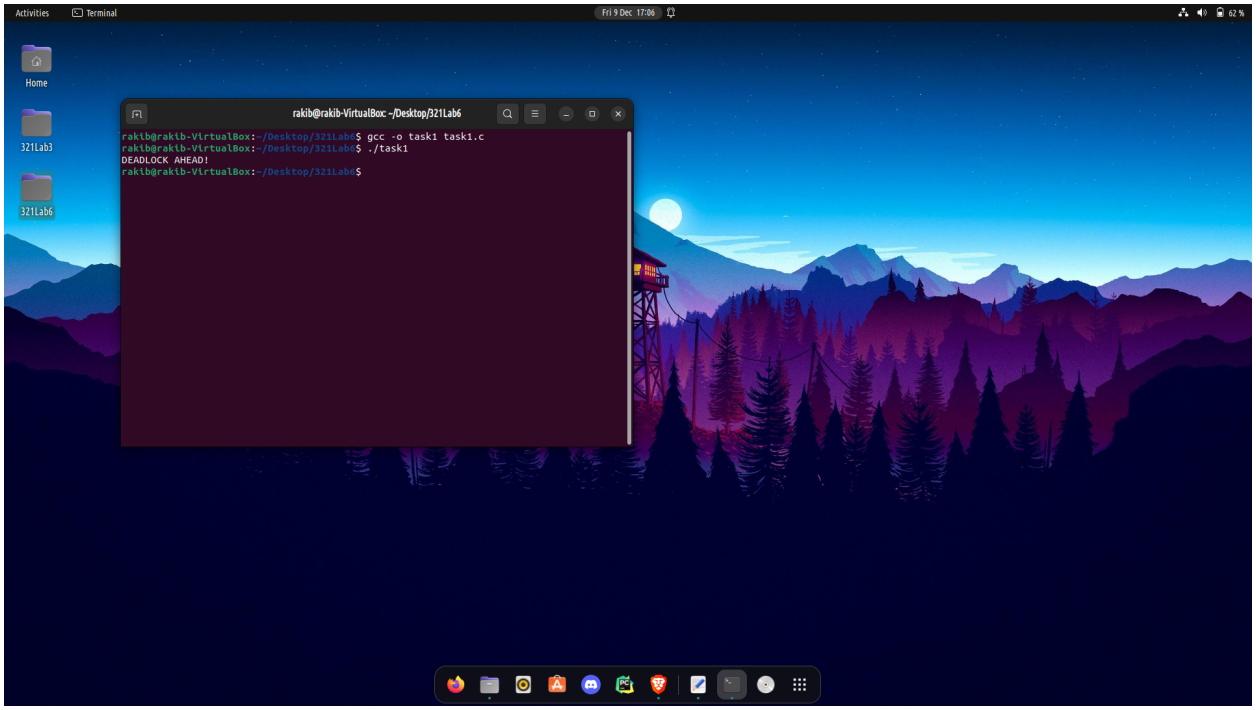
Task-1:

```
#include <stdbool.h>
#include <stdio.h>

void deadlock(int N, int M, int allocation[N][M], int max[N][M],
                    int avail[M]) {
    bool flag = true;
    int rem = N;
    bool comp[N];
    for (int i = 0; i < N; i++) {
        comp[i] = false;
    }
    while (flag && rem > 0) {
        flag = false;
        for (int i = 0; i < N; i++) {
            if (comp[i] == true) {
                continue;
            }

            bool resource = true;
            for (int j = 0; j < M; j++) {
                int need = max[i][j] - allocation[i][j];
                if (need > allocation[i][j]) {
                    resource = false;
                    break;
                }
            }
            if (resource) {
                rem--;
                flag = false;
                for (int j = 0; j < M; j++) {
                    avail[j] += allocation[i][j];
                }
            }
        }
    }
    if (rem == 0) {
        printf("SAFE HERE!\n");
    } else {
        printf("DEADLOCK AHEAD!\n");
```

```
    }
}

int main(int argc, char *argv[]) {

    // given inputs
    int n = 5; // Number of processes
int m = 4; // Number of resources
int alloc[5][4] = { { 0, 1, 0, 3 }, // P0   // Allocation Matrix
        { 2, 0, 0, 0 }, // P1
        { 3, 0, 2, 0 }, // P2
        { 2, 1, 1, 5 }, // P3
        { 0, 0, 2, 2 } }; // P4

int max[5][4] = { { 6, 4, 3, 4 }, // P0     // MAX Matrix
      { 3, 2, 2, 1 }, // P1
      { 9, 1, 2, 6 }, // P2
      { 2, 2, 2, 8 }, // P3
      { 4, 3, 3, 7 } }; // P4

int total[4] = {10, 5, 7, 11};      //Total resources
int avail[4];

    // function call
    deadlock(n, m, alloc, max, avail);
    return 0;
}
```



Task-2:

```
#include <stdbool.h>
#include <stdio.h>

void safe_seq(int N, int M, int allocation[N][M], int max[N][M], int
avail[M]) {
    int rem = N;
    bool comp[N];
    for (int i = 0; i < N; i++) {
        comp[i] = false;
    }

    int safeSeq[N];
    int index = 0;
    bool flag = true;

    while (flag && rem > 0) {
        flag = false;
        for (int i = 0; i < N; i++) {
            if (comp[i] == true) {
                continue;
```

```
            }
            bool enough_res = true;
            for (int j = 0; j < M; j++) {
                int need = max[i][j] - allocation[i][j];
                if (need > avail[j]) {
                    enough_res = false;
                    break;
                }
            }

            if (enough_res) {
                flag = true;
                comp[i] = true;
                safeSeq[index] = i;
                index++;
                rem--;
                for (int j = 0; j < M; j++) {
                    avail[j] += allocation[i][j];
                }
            }
        }
    }
    if (rem == 0) {
        printf("Safe sequence: ");
        for (int i = 0; i < N; i++) {
            printf("P%d", safeSeq[i]);
            if (i == N - 1) {
                printf("\n");
            } else {
                printf(" --> ");
            }
        }
    } else {
        printf("Deadlock Ahead!\n");
    }
}

int main(int argc, char *argv[]) {

    int n = 6; // Number of processes
```

```
    int m = 4; // Number of resources
    int alloc[6][4] = {
        {0, 1, 0, 3},
        {2, 0, 0, 3},
        {3, 0, 2, 0},
        {2, 1, 1, 5},
        {0, 0, 2, 2},
        {1, 2, 3, 1},
    };
    int max[6][4] = {
        {6, 4, 3, 4},
        {3, 2, 2, 4},
        {9, 1, 2, 6},
        {2, 2, 2, 8},
        {4, 3, 3, 7},
        {6, 2, 6, 5},
    };
    int avail[4] = {2, 2, 2, 1};
    safe_seq(n, m, alloc, max, avail);

    return 0;
}
```