# BLG312E HW3 REPORT

Berkay Sancı - 504231561

With respect to the information given about the homework, that the code will be tested with different input files and number of resources and processes will be the same, I declared the necessary arrays dynamically since the size could vary with different input files. I created a 'Process' which has 'id', 'allocations' and 'requests' as member variables.

```c
typedef struct {
    int id;
    int *allocations;
    int *requests;
} Process;
```

After that, I created the necessary functions. I assigned the number of resources by reading the elements in the 'resources.txt' file. I did this because it may differ for different input files.

```c
int count_resources(const char *filename) {
    FILE *file = fopen(filename, "r");
    if (!file) {
        perror("Failed to open resources file");
        exit(EXIT_FAILURE);
    }

    char line[256];
    if (fgets(line, sizeof(line), file) == NULL) {
        perror("Failed to read resources file");
        fclose(file);
        exit(EXIT_FAILURE);
    }
    fclose(file);

    // Count the number of elements in the line
    int count = 0;
    char *token = strtok(line, " ");
    while (token != NULL) {
        count++;
        token = strtok(NULL, " ");
    }
    return count;
}
```

With this function, I was able to assign number of processes and number of resources. After that, I wrote the functions in order to read values from 'resources.txt', 'allocations.txt' and 'requests.txt'. With these functions, I read these values and assigned it to the corresponding process.

After that, I wrote the function 'detectdeadlock' in which Banker's algorithm is implemented. Inside this function, 2D matrices 'allocatedresources' and 'maxresources' are declared in order to show allocations of each resource

for each process and the maximum value of every resource for each process. Another matrix 'available' is used to keep available amount for each resource. There is also 'finished' matrix which is used to keep track of if a process is finished.

Allocated resources matrix is filled with the allocation value of each process. Max resources matrix is filled with the 'request+allocation' value of each process. Then available values are found by subtracting every allocation of a resource from total resources. Finished matrix is initialized with 0.

Then 2D 'need' matrix is created and it is filled with max - allocated value for each process and resource. There is also 'running order' matrix which is used to keep the execution of processes which will be printed. Then the Banker's algorithm is implemented. For a process, if the available resources are bigger than needed, it is added to running order. After that process finisihes, its resources are added to available resources.

At the end, if there is a process that is not finished, then it means there is deadlock. Remaining resources that are not in running order are the causes of deadlock. These informations are printed.

After compiling and running the code, these results are obtained:

```
Information for process : P1:
Allocated resources : R1:3 R2:0 R3:1 R4:1 R5:0
Resource request : R1:0 R2:1 R3:7 R4:0 R5:1

Information for process : P2:
Allocated resources : R1:1 R2:1 R3:0 R4:0 R5:0
Resource request : R1:0 R2:0 R3:1 R4:0 R5:3

Information for process : P3:
Allocated resources : R1:0 R2:3 R3:0 R4:0 R5:0
Resource request : R1:2 R2:2 R3:0 R4:0 R5:1

Information for process : P4:
Allocated resources : R1:1 R2:0 R3:0 R4:0 R5:0
Resource request : R1:1 R2:0 R3:1 R4:0 R5:2

Information for process : P5:
Allocated resources : R1:0 R2:1 R3:4 R4:0 R5:0
Resource request : R1:3 R2:1 R3:0 R4:1 R5:1
```

```
Running order for processes: P2 P4 P3
There is a deadlock. P1 and P5 are the cause of deadlock.
```