Assignment No: 4

Name: Kedar Kale

Roll No: CETTY129

Topic: Banker's Algorithm

Code:

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

int main() {
   int curr[5][5];
   int max_claim[5][5];
   int avl[5];
   int alloc[5] = {0, 0, 0, 0, 0};
   int max_res[5];
   int running[5];

   int i, j, exec, r, p;
   int count = 0;
   bool safe = false;

   printf("\nEnter the number of resources: ");
   scanf("%d", &r);

   printf("\nEnter the number of processes: ");
   scanf("%d", &p);
   for (i = 0; i < p; i++) {
      running[i] = 1;
      count++;
   }

   printf("\nEnter Claim Vector: ");
   for (i = 0; i < r; i++)
      scanf("%d", &max_res[i]);

   printf("\nEnter Allocated Resource Table: ");
   for (i = 0; i < p; i++) {
      for (j = 0; j < r; j++)
         scanf("%d", &curr[i][j]);
   }

   printf("\nEnter Maximum Claim table: ");
   for (i = 0; i < p; i++) {
      for (j = 0; j < r; j++)
         scanf("%d", &max_claim[i][j]);
   }
```

```
printf("\nThe Claim Vector is: ");
for (i = 0; i < r; i++)
   printf("%d ", max_res[i]);

printf("\nThe Allocated Resource Table:\n");
for (i = 0; i < p; i++) {
   for (j = 0; j < r; j++)
      printf("\t%d", curr[i][j]);
   printf("\n");
}

printf("\nThe Maximum Claim Table:\n");
for (i = 0; i < p; i++) {
   for (j = 0; j < r; j++)
      printf("\t%d", max_claim[i][j]);
   printf("\n");
}

for (i = 0; i < p; i++)
   for (j = 0; j < r; j++)
      alloc[j] += curr[i][j];

printf("\nAllocated resources: ");
for (i = 0; i < r; i++)
   printf("%d ", alloc[i]);
for (i = 0; i < r; i++)
   avl[i] = max_res[i] - alloc[i];

printf("\nAvailable resources: ");
for (i = 0; i < r; i++)
   printf("%d ", avl[i]);
printf("\n");

while (count != 0) {
   safe = false;
   for (i = 0; i < p; i++) {
      if (running[i]) {
         exec = 1;
         for (j = 0; j < r; j++) {
            if (max_claim[i][j] - curr[i][j] > avl[j]) {
               exec = 0;
               break;
            }
         }

         if (exec) {
            printf("\nProcess%d is executing.\n", i + 1);
```

```
                running[i] = 0;
                count--;
                safe = true;
                for (j = 0; j < r; j++)
                   avl[j] += curr[i][j];
                break;
             }
          }
       }

       if (!safe) {
          printf("\nThe processes are in unsafe state.");
          break;
       }

       if (safe)
          printf("\nThe process is in safe state.");

       printf("\nAvailable vector: ");
       for (i = 0; i < r; i++)
          printf("%d ", avl[i]);
    }

    return 0;
}
```

Output:

Enter the number of resources: 4

Enter the number of processes: 5

Enter Claim Vector: 8 5 9 7

Enter Allocated Resource Table: 2 0 1 1 0 1 2 1 4 0 0 3 0 2 1 0 1 0 3 0

Enter Maximum Claim table: 3 2 1 4 0 2 5 2 5 1 0 5 1 5 3 0 3 0 3 3

The Claim Vector is: 8 5 9 7

The Allocated Resource Table:

2 0 1 1

0 1 2 1

4 0 0 3

0 2 1 0

1 0 3 0

The Maximum Claim Table:

3 2 1 4

0 2 5 2

5 1 0 5

1 5 3 0

3 0 3 3

Allocated resources: 7 3 7 5

Available resources: 1 2 2 2

Process3 is executing.

The process is in safe state.

Available vector: 5 2 2 5

Process1 is executing.

The process is in safe state.

Available vector: 7 2 3 6

Process2 is executing.

The process is in safe state.

Available vector: 7 3 5 7

Process4 is executing.

The process is in safe state.

Available vector: 7 5 6 7

Process5 is executing.

The process is in safe state.

Available vector: 8 5 9 7