13. Design a C program Illustrate the deadlock avoidance concept by simulating Banker's algorithm using C.

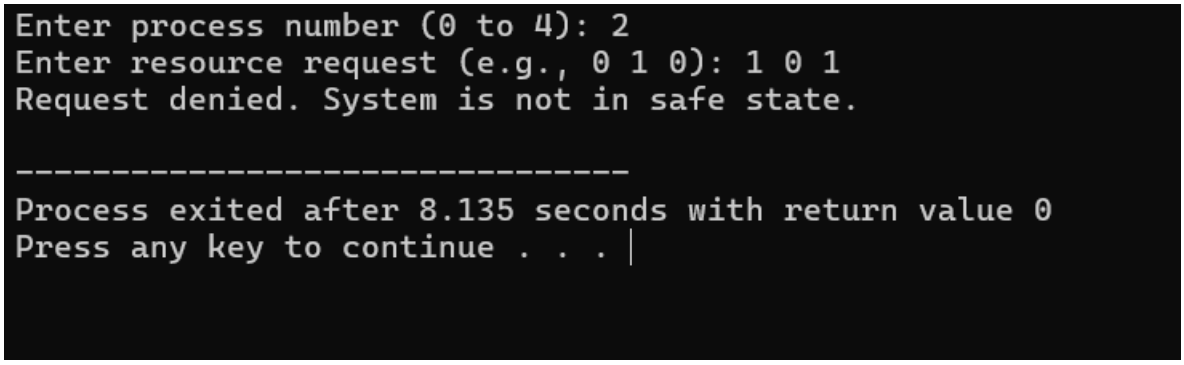
PROGRAM :

```
#include <stdio.h>

#define MAX_PROCESSES 5

#define MAX_RESOURCES 3

int is_safe();

int available[MAX_RESOURCES] = {3, 3, 2}; // Available instances of each resource

int maximum[MAX_PROCESSES][MAX_RESOURCES] = {{7, 5, 3}, {3, 2, 2}, {9, 0, 2}, {2, 2, 2}, {4, 3, 3}};

int allocation[MAX_PROCESSES][MAX_RESOURCES] = {{0, 1, 0}, {2, 0, 0}, {3, 0, 2}, {2, 1, 1}, {0, 0, 2}};

int request_resources(int process_num, int request[]) {

// Check if request can be granted

for (int i = 0; i < MAX_RESOURCES; i++) {

if (request[i] > available[i] || request[i] > maximum[process_num][i] - allocation[process_num][i])

return 0; // Request cannot be granted

}

// Try allocating resources temporarily

for (int i = 0; i < MAX_RESOURCES; i++) {

available[i] -= request[i];

allocation[process_num][i] += request[i];

// Update maximum and need matrix if request is granted

maximum[process_num][i] -= request[i];

}

// Check if system is in safe state after allocation if

if(is_safe()) {

return 1; // Request is granted

}

else {

// Roll back changes if not safe
```

```
for (int i = 0; i < MAX_RESOURCES; i++) {
available[i] += request[i];
allocation[process_num][i] -= request[i];
maximum[process_num][i] += request[i];
}
return 0; // Request is denied
}
}
int is_safe() {
int work[MAX_RESOURCES];
int finish[MAX_PROCESSES] = {0};
// Initialize work array
for (int i = 0; i < MAX_RESOURCES; i++) {
work[i] = available[i];
}
// Check if processes can finish
int count = 0;
while (count < MAX_PROCESSES) {
int found = 0;
for (int i = 0; i < MAX_PROCESSES; i++) {
if (finish[i] == 0) {
int j;
for (j = 0; j < MAX_RESOURCES; j++) {
if (maximum[i][j] - allocation[i][j] > work[j])
break;
}
if (j == MAX_RESOURCES) {
// Process can finish, update work and mark as finished
for (int k = 0; k < MAX_RESOURCES; k++) {
work[k] += allocation[i][k];
```

```
}
finish[i] = 1;
found = 1;
count++;
}
}
}
if (found == 0) {
return 0; // No process can finish, not safe state
}
}
return 1; // All processes can finish, safe state
}
int main() {
int process_num, request[MAX_RESOURCES];
printf("Enter process number (0 to 4): "); scanf("%d", &process_num);
printf("Enter resource request (e.g., 0 1 0): ");
for (int i = 0; i < MAX_RESOURCES; i++) {
scanf("%d", &request[i]);
}
if (request_resources(process_num, request)) {
printf("Request granted.\n");
} else {
printf("Request denied. System is not in safe state.\n");
}
return 0;
}
```

OUTPUT :


Enter process number (0 to 4): 2
Enter resource request (e.g., 0 1 0): 1 0 1
Request denied. System is not in safe state.

Process exited after 8.135 seconds with return value 0
Press any key to continue . . .