5. **Illustrate the deadlock avoidance concept by simulating Banker’s algorithm with C.**

**int main()**

**{**

**int count = 0, m, n, process, temp, resource; int allocation\_table[5] = {0, 0, 0, 0, 0}; int available[5], current[5][5], maximum\_claim[5][5]; int maximum\_resources[5], running[5], safe\_state = 0; printf("\nEnter The Total Number Of Processes:\t"); scanf("%d", &process); for(m=0;m<process;m++)**

**{ running[m]=1; count++;**

**}**

**printf("\nEnter The Total Number Of Resources To Allocate:\t"); scanf("%d",&resource);**

**printf("\nEnter The Claim Vector:\t"); for(m=0;m<resource;m++)**

**{**

**scanf("%d",&maximum\_resources[m]);**

**}**

**printf("\nEnter Allocated Resource Table:\n"); for(m=0;m<process;m++)**

**{**

**for(n=0;n<resource;n++)**

**{**

**scanf("%d",&current[m][n]);**

**}**

**}**

**printf("\nEnter The Maximum Claim Table:\n");for(m=0;m<process;m++)**

**{**

**for(n=0;n<resource;n++)**

**{**

**scanf("%d",&maximum\_claim[m][n]);**

**} }**

**printf("\nThe Claim Vector \n");**

**for(m=0;m<resource;m++)**

**{printf("\t%d ",maximum\_resources[m]);**

**}**

**printf("\n The Allocated Resource Table\n"); for(m=0;m<process;m++)**

**{for(n=0;n<resource;n++)**

**{printf("\t%d",current[m][n]);**

**} printf("\n");**

**}printf("\nThe Maximum Claim Table \n"); for(m=0;m<process;m++)**

**{for(n=0;n<resource;n++)**

**{printf("\t%d",maximum\_claim[m][n]);**

**} printf("\n");**

**}for(m=0;m<process;m++)**

**{for(n=0;n<resource;n++)**

**{allocation\_table[n]=allocation\_table[n]+current[m][n];**

**}}**

**printf("\nAllocated Resources \n"); for(m=0;m<resource;m++)**

**{printf("\t%d",allocation\_table[m]);**

**}for(m=0;m<resource;m++)**

**{**

**available[m]=maximum\_resources[m]-allocation\_table[m];**

**}**

**printf("\nAvailable Resources:");**

**for(m=0;m<resource;m++)**

**{**

**printf("\t%d",available[m]);**

**} printf("\n"); while(count!=0)**

**{ safe\_state=0;**

**for(m=0;m<process;m++)**

**{**

**if(running[m])**

**{ temp=1;**

**for(n=0;n<resource;n++)**

**{**

**if(maximum\_claim[m][n]-current[m][n]>available[n])**

**{ temp=0; break;**

**}**

**} if(temp)**

**{**

**printf("\nProcess %d Is In Execution \n", m + 1); running[m]=0; count--; safe\_state=1;**

**for(n=0;n<resource;n++)**

**{available[n]=available[n]+current[m][n];**

**} break;**

**}}}if(!safe\_state)**

**{printf("\nThe Processes Are In An Unsafe State \n"); break; } else**

**{printf("\nThe Process Is In A Safe State \n"); printf("\nAvailable Vector\n");**

**for(m=0;m<resource;m++)**

**{printf("\t%d",available[m]);**

**} printf("\n");**

**}}**

**}**

