Student Name:	Aditya.Rajesh.Wanwade
Roll No:	C2-27
Practical No:	6
Aim:	Write a C program to simulate Bankers Algorithm for Deadlock Avoidance.

Aim: To write a C program to implement bankers' algorithm for dead lock avoidance:

```
#include <stdio.h>
struct process
{
   int allocation[3];
   int max[3];
   int need[3];
   int finish;
} p[10];
int main()
{
   int n, i, I, j, avail[3], work[3], flag, count = 0, sequence[10], k = 0;
   printf("\nEnter the number of process:");
   scanf("%d", &n);
   for (i = 0; i < n; i++)
   {
      printf("\nEnter the %dth process allocated resources:", i);
      scanf("%d%d%d", &p[i].allocation[0], &p[i].allocation[1], &p[i].allocation[2]);
      printf("\nEnter the %dth process maximum resources:", i);
      scanf("%d%d%d", &p[i].max[0], &p[i].max[1], &p[i].max[2]);
      p[i].finish = 0;
      p[i].need[0] = p[i].max[0] - p[i].allocation[0];
      p[i].need[1] = p[i].max[1] - p[i].allocation[1];
      p[i].need[2] = p[i].max[2] - p[i].allocation[2];
   }
   printf("\nEnter the available vector:");
   scanf("%d%d%d", &avail[0], &avail[1], &avail[2]);
   for (i = 0; i < 3; i++)
      work[i] = avail[i];
```

```
    while (count != n)
    {
        count = 0;
        for (i = 0; i < n; i++)
        {
            flag = 1;
            if (p[i].finish == 0)
                if (p[i].need[0] <= work[0])
                    if (p[i].need[1] <= work[1])
                        if (p[i].need[2] <= work[2])
                        {
                            for (j = 0; j < 3; j++)
                                work[j] += p[i].allocation[j];
                            p[i].finish = 1;
                            sequence[k++] = i;
                            flag = 0;
                        }
            if (flag == 1)
                count++;
        }
    }
    count = 0;
    for (i = 0; i < n; i++)
        if (p[i].finish == 1)
            count++;
    printf("\n The safe sequence is:\t");
    if (count++ == n)
        for (i = 0; i < k; i++)
            printf("%d\n", sequence[i]);
    else
        printf("SYSTEM IS NOT IN A SAFE STATE \n\n");
    return 0;
}
```

OUTPUT: (All the test cases are included):

```
┌──(kali㉿kali)-[~/C27/lab6]
└─$ vi lab6.c

┌──(kali㉿kali)-[~/C27/lab6]
└─$ gcc lab6.c

┌──(kali㉿kali)-[~/C27/lab6]
└─$ ./a.out
Enter the number of process:3

Enter the 0th process allocated resources:1 2 3

Enter the 0th process maximum resources:4 5 6

Enter the 1th process allocated resources:3 4 5

Enter the 1th process maximum resources:6 7 8

Enter the 2th process allocated resources:1 2 3

Enter the 2th process maximum resources:3 4 5

Enter the available vector:10 12 11

 The safe sequence is:  0
1
2

┌──(kali㉿kali)-[~/C27/lab6]
└─$
```

Result: Thus the program was executed and verified successfully.