WORST FIT ALGORITHM

AIM:

TO WRITE A PROGARM FOR THE WORST FIT ALGORITHM USING C LANGUAGE .

The worst fit memory allocation scheme, the operating system searches for free memory blocks demanded by the operating system.

An empty block is assigned to the processes as soon as the CPU identifies it.

The scheme is also said as the worst fit memory management scheme as sometimes a process is allocated a memory block which is much larger to the actual demand resulting in a huge amount of wasted memory.

ALGORITHM:

```
1- Input memory blocks and processes with sizes.
2- Initialize all memory blocks as free.
3- Start by picking each process and find the
   maximum block size that can be assigned to
   current process i.e., find max(bockSize[1],
   blockSize[2],.....blockSize[n]) >
   processSize[current], if found then assign
   it to the current process.
5- If not then leave that process and keep checking
   the further processes.
```

OR

- **Step 1:** Input memory block with a size.
- **Step 2:** Input process with size.
- **Step 3:** Initialize by selecting each process to find the maximum block size that can be assigned to the current process.
- **Step 4:** If the condition does not fulfill, they leave the process.
- **Step 5:** If the condition is not fulfilled, then leave the process and check for the next process.
- **Step 6:** Stop.

Write a C Program yo implement Worst Fit algorithm?

Sample Input :

3 (No.of Blocks)

3 (No.of Process)

100 (Size of Block 1)

500 (Size of Block 2)

200 (Size of Block 3)

20 (Size of process 1)

300 (Size of Process 2)

150 (Size of Process 3)

3 (No.of Process)

100 (Size of Block 1)

500 (Size of Block 2)

200 (Size of Block 3)

20 (Size of process 1)

300 (Size of Process 2)

150 (Size of Process 3)

Expected Output:

Process_no:	Process_size :	Block_no:	Block_size:	Fragment
1	20	2	500	480
2	300	0	1	0
3	150	3	200	50

For example:

Test	Input	Result				
T1	3 3 200 400 600 100 150 50	Process_no: 1 2 3	Process_size : 100 150 50	Block_no: 3 2 1	Block_size: 600 400 200	Fragment 500 250 150

PROGRAM:

```c
#include<stdio.h>
#define max 25
int main()
{
    int frag[max], b[max], f[max], i, j, nb, nf, temp, highest = 0;
    static int bf[max], ff[max];
    scanf("%d", &nb);
    scanf("%d", &nf);
    for(i = 1; i <= nb; i++)
    {
        scanf("%d", &b[i]);
    }
    for(i = 1; i <= nb; i++)
    {
        scanf("%d", &f[i]);
    }
    for(i = 1; i <= nf; i++)
    {
        for(j = 1; j <= nb; j++)
        {
            if(bf[j]!=1)
            {
                temp = b[j] - f[i];
                if(temp >= 0)
                if(highest<temp)
                {
                    ff[i]=j;
                    highest = temp;
                }
            }
        }
        frag[i]=highest;
        bf[ff[i]]=1;
        highest=0;
    }
    printf("Process_no:\tProcess_size :\tBlock_no:\tBlock_size:\tFragment
    for(i = 1; i <= nf; i++)
    printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", i, f[i], ff[i], b[ff[i]], fra
    return 0;
}
```

```
20  {
21      if(bf[j]!=1)
22      {
23          temp = b[j] - f[i];
24          if(temp >= 0)
25          if(highest<temp)
26          {
27              ff[i]=j;
28              highest = temp;
29          }
30      }
31  }
32  frag[i]=highest;
33  bf[ff[i]]=1;
34  highest=0;
35
36 intf("Process_no:\tProcess_size :\tBlock_no:\tBlock_size:\tFragment");
37 or(i = 1; i <= nf; i++)
38 intf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", i, f[i], ff[i], b[ff[i]], frag[i]);
39 eturn 0;
40
```

OUTPUT:

	Test	Input	Expected	Got
✔	T1	3 3 200 400 600 100 150 50	Process_no:\tProcess_size :\tBlock_no:\tBlock_size:\tFragment 1\t\t100\t\t3\t\t600\t\t500 2\t\t150\t\t2\t\t400\t\t250 3\t\t50\t\t1\t\t200\t\t150	Process_no:\ 1\t\t100\t\t 2\t\t150\t\t 3\t\t50\t\t1
✔	T2	3 3 700 200 400 100 150 50	Process_no:\tProcess_size :\tBlock_no:\tBlock_size:\tFragment 1\t\t100\t\t1\t\t700\t\t600 2\t\t150\t\t3\t\t400\t\t250 3\t\t50\t\t2\t\t200\t\t150	Process_no:\ 1\t\t100\t\t 2\t\t150\t\t 3\t\t50\t\t2

Passed all tests! ✔

RESULT:

THE GIVEN PROGRAM FOR WORST FIT ALGORITHM USING C LANGUAGE WAS SUCCESSFULLY EXECUTED.