BEST FIT ALGORITHM.

AIM:

TO WRITE A PROGRAM FOR BEST FIT ALGORITHM USING C LANGUAGE.

The best fit deals with allocating the smallest free partition which meets the requirement of the requesting process.

This algorithm first searches the entire list of free partitions and considers the smallest hole that is adequate.

It then tries to find a hole which is close to actual process size needed.

OR

the best fit memory allocation scheme, the operating system searches for the empty memory block.

When the operating system finds the memory block with minimum wastage of memory, it is allocated to the process.

This scheme is considered as the best approach as it results in most optimized memory allocation. However, finding the best fit memory allocation may be time-consuming.

ALGORITHM:

Step1. Enter the memory blocks with size.

Step2. Enter the process blocks with size.

Step3. Set all the memory blocks as free.

Step4. Start by picking up each process

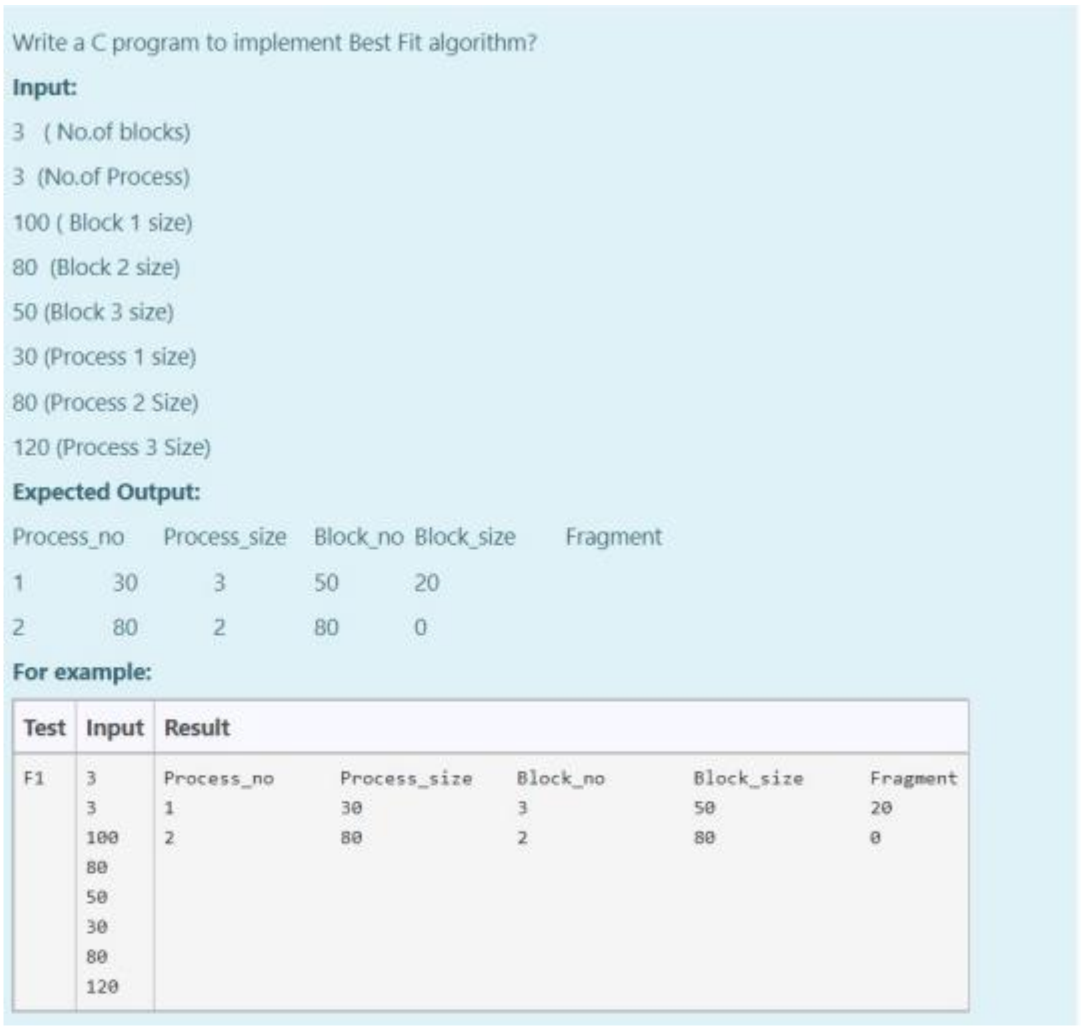
Step5. Find the minimum block size that is best to assign to the current process.

Step6. If the best fit memory size is found, it is allocated to the process.

Step7. If the memory block and memory demand do not match, leave the process and search for another process.

OR

1. Get no. of Processes and no. of blocks.
2. After that get the size of each block and process requests.
3. Then select the best memory block that can be allocated using the above definition.
4. Display the processes with the blocks that are allocated to a respective process.
5. Value of Fragmentation is optional to display to keep track of wasted memory.
6. Stop.



Write a C program to implement Best Fit algorithm?

Input:

3 (No.of blocks)

3 (No.of Process)

100 (Block 1 size)

80 (Block 2 size)

50 (Block 3 size)

30 (Process 1 size)

80 (Process 2 Size)

120 (Process 3 Size)

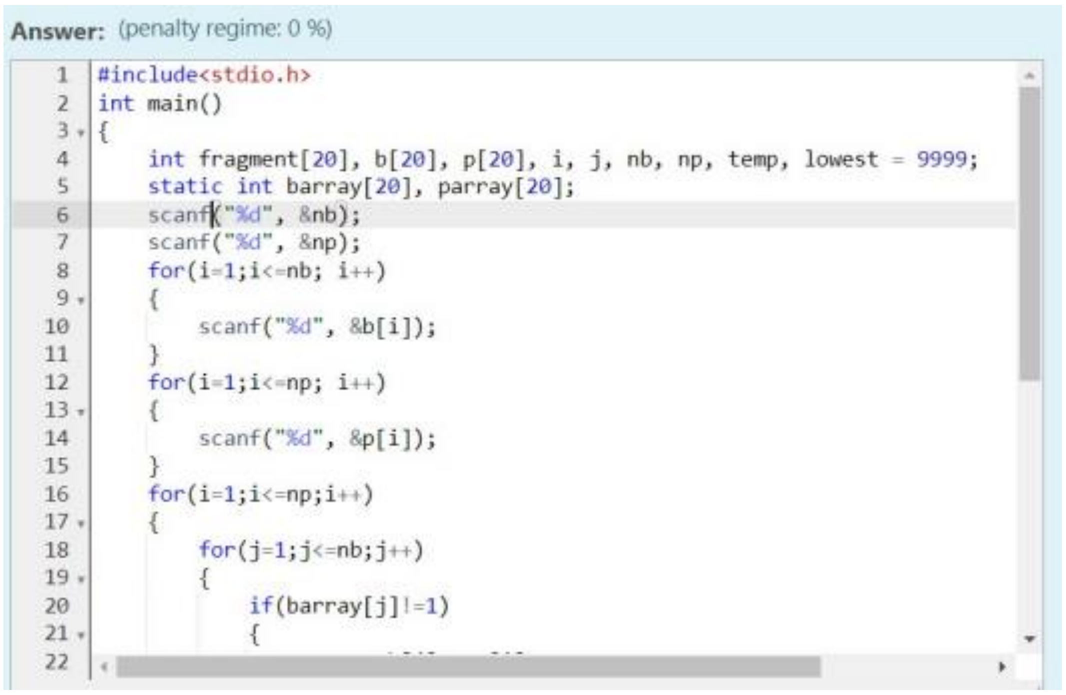
Expected Output:

```
Process_no    Process_size   Block_no  Block_size     Fragment
1          30        3         50        20
2          80        2         80        0
```

For example:

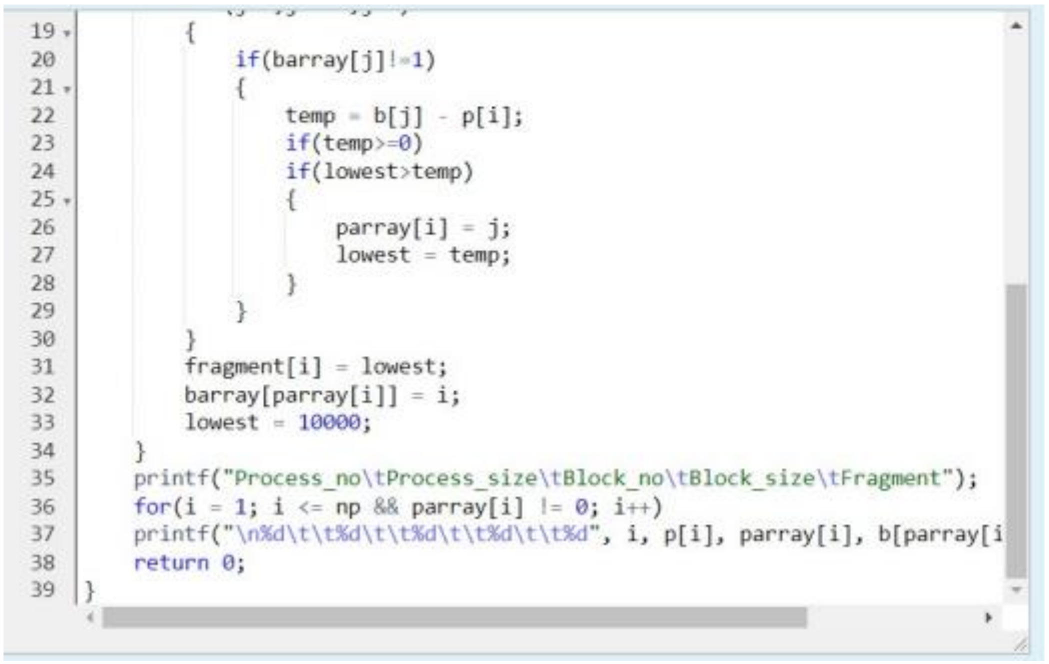
Test	Input	Result				
F1	3	Process_no	Process_size	Block_no	Block_size	Fragment
	3	1	30	3	50	20
	100	2	80	2	80	0
	80					
	50					
	30					
	80					
	120					

PROGRAM:

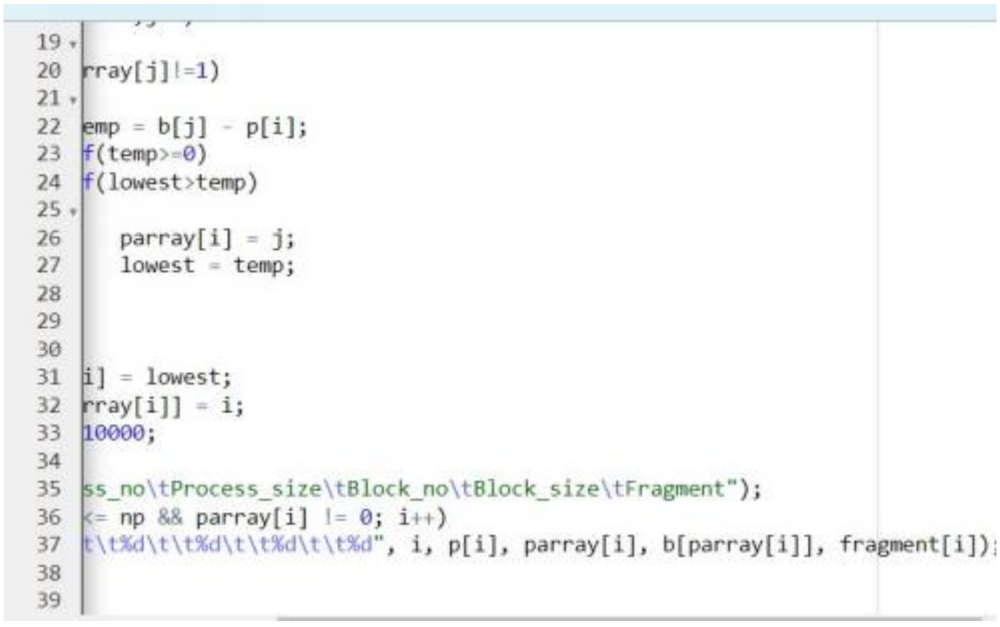


Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main()
{
    int fragment[20], b[20], p[20], i, j, nb, np, temp, lowest = 9999;
    static int barray[20], parray[20];
    scanf("%d", &nb);
    scanf("%d", &np);
    for(i=1;i<=nb; i++)
    {
        scanf("%d", &b[i]);
    }
    for(i=1;i<=np; i++)
    {
        scanf("%d", &p[i]);
    }
    for(i=1;i<=np;i++)
    {
        for(j=1;j<=nb;j++)
        {
            if(barray[j]!=1)
            {
```

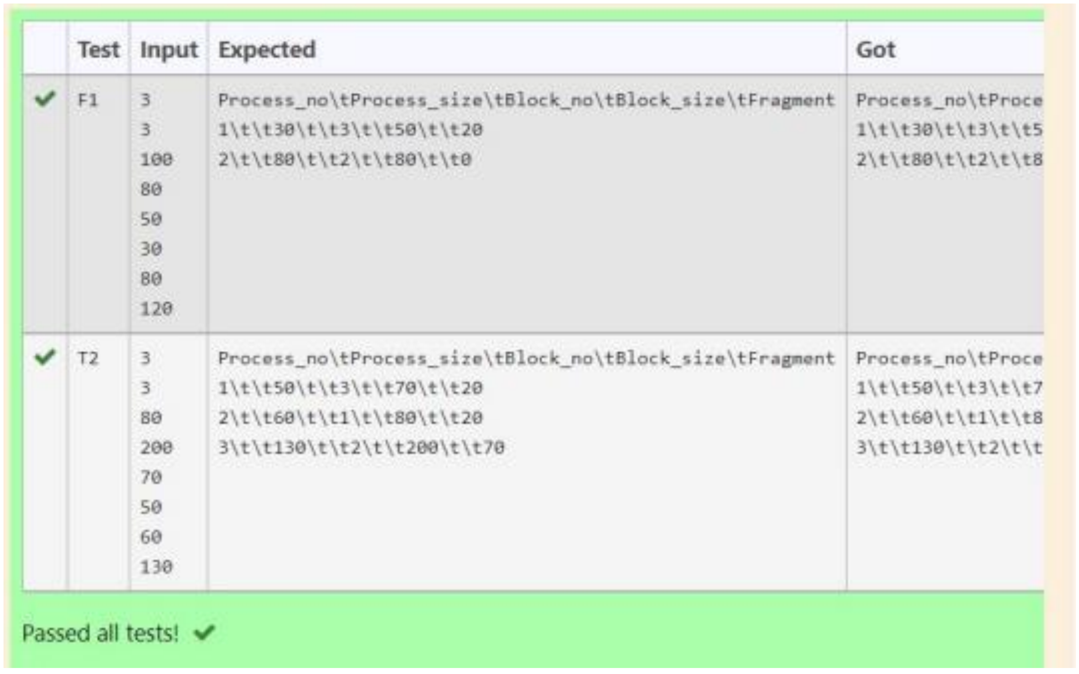


```c
        {
            if(barray[j]!=1)
            {
                temp = b[j] - p[i];
                if(temp>=0)
                if(lowest>temp)
                {
                    parray[i] = j;
                    lowest = temp;
                }
            }
        }
        fragment[i] = lowest;
        barray[parray[i]] = i;
        lowest = 10000;
    }
    printf("Process_no\tProcess_size\tBlock_no\tBlock_size\tFragment");
    for(i = 1; i <= np && parray[i] != 0; i++)
    printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", i, p[i], parray[i], b[parray[i
    return 0;
}
```



```
rray[j]!=1)

emp = b[j] - p[i];
f(temp>=0)
f(lowest>temp)

   parray[i] = j;
   lowest = temp;



i] = lowest;
rray[i]] = i;
10000;

ss_no\tProcess_size\tBlock_no\tBlock_size\tFragment");
<= np && parray[i] != 0; i++)
t\t%d\t\t%d\t\t%d\t\t%d", i, p[i], parray[i], b[parray[i]], fragment[i]);
```

OUTPUT:



	Test	Input	Expected	Got
✓	F1	3 3 100 80 50 30 80 120	Process_no\tProcess_size\tBlock_no\tBlock_size\tFragment 1\t\t30\t\t3\t\t50\t\t20 2\t\t80\t\t2\t\t80\t\t0	Process_no\tProce 1\t\t30\t\t3\t\t5 2\t\t80\t\t2\t\t8
✓	T2	3 3 80 200 70 50 60 130	Process_no\tProcess_size\tBlock_no\tBlock_size\tFragment 1\t\t50\t\t3\t\t70\t\t20 2\t\t60\t\t1\t\t80\t\t20 3\t\t130\t\t2\t\t200\t\t70	Process_no\tProce 1\t\t50\t\t3\t\t7 2\t\t60\t\t1\t\t8 3\t\t130\t\t2\t\t\t

Passed all tests! ✓

RESULT:

BEST FIT ALGORITHM WAS SUCCESSFULLY IMPLEMENT USING C LANGUAGE.