Name: P.MALREDDY

Reg-No: 192372015

21.Develop a C program to implement the worst fit algorithm of memory management.

Aim:

To implement the **Worst Fit** memory allocation algorithm in C for managing memory allocation to processes, ensuring the largest free memory block is allocated to a process.

Algorithm:

1. Input the size of memory blocks and processes.
2. For each process:
    - Find the largest memory block that can fit the process.
    - Allocate the block to the process.
    - Reduce the block's size by the process's size.
3. If no suitable block is found, the process remains unallocated.
4. Display the allocation results.

Procedure:

1. Take input for the sizes of memory blocks and processes.
2. Traverse the memory blocks to find the largest block for each process.
3. Update memory block size and allocation details.
4. Print the allocation results for each process.

Code:

```
#include <stdio.h>



int main() {

    int blocks[10], processes[10], allocation[10];

    int nBlocks, nProcesses;



    printf("Enter number of memory blocks: ");

    scanf("%d", &nBlocks);

    printf("Enter sizes of memory blocks: ");
```

```
for (int i = 0; i < nBlocks; i++) scanf("%d", &blocks[i]);


printf("Enter number of processes: ");

scanf("%d", &nProcesses);

printf("Enter sizes of processes: ");

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

    scanf("%d", &processes[i]);

    allocation[i] = -1;

}


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

    int worstIdx = -1;

    for (int j = 0; j < nBlocks; j++) {

        if (blocks[j] >= processes[i]) {

            if (worstIdx == -1 || blocks[j] > blocks[worstIdx])

                worstIdx = j;

        }

    }

    if (worstIdx != -1) {

        allocation[i] = worstIdx;

        blocks[worstIdx] -= processes[i];

    }

}
```

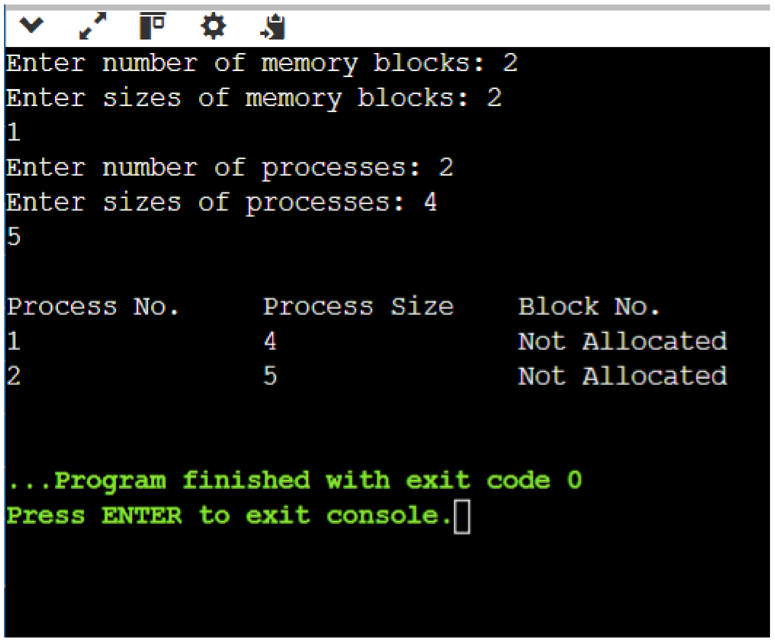
```
    printf("\nProcess No.\tProcess Size\tBlock No.\n");
    for (int i = 0; i < nProcesses; i++) {
        printf("%d\t\t%d\t\t", i + 1, processes[i]);
        if (allocation[i] != -1)
            printf("%d\n", allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }

    return 0;
}
```

Result:

his demonstrates the **Worst Fit** algorithm where the process is allocated the largest block that fits, or remains unallocated if no suitable block is available.

Output:



```
Enter number of memory blocks: 2
Enter sizes of memory blocks: 2
1
Enter number of processes: 2
Enter sizes of processes: 4
5

Process No.     Process Size    Block No.
1               4               Not Allocated
2               5               Not Allocated


...Program finished with exit code 0
Press ENTER to exit console.
```