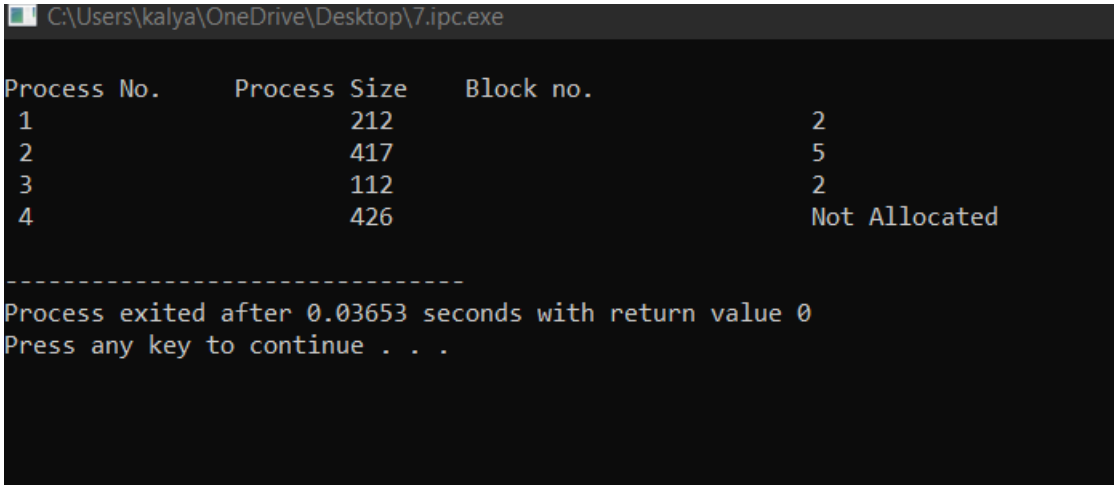
10. Construct a C program for implementation of memory allocation using first fit strategy.

```c
#include<stdio.h>

// Function to allocate memory to
// blocks as per First fit algorithm
void firstFit(int blockSize[], int m, int processSize[], int n)
{
    int i, j;
    // Stores block id of the
    // block allocated to a process
    int allocation[n];

    // Initially no block is assigned to any process
    for(i = 0; i < n; i++)
    {
        allocation[i] = -1;
    }

    // pick each process and find suitable blocks
    // according to its size ad assign to it
    for (i = 0; i < n; i++)        //here, n -> number of processes
    {
        for (j = 0; j < m; j++)        //here, m -> number of blocks
        {
            if (blockSize[j] >= processSize[i])
            {
                // allocating block j to the ith process
                allocation[i] = j;

                // Reduce available memory in this block.
                blockSize[j] -= processSize[i];

                break;    //go to the next process in the queue
            }
        }
    }

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

```c
// Driver code
int main()
{
    int m;    //number of blocks in the memory
      int n;  //number of processes in the input queue
      int blockSize[] = {100, 500, 200, 300, 600};
    int processSize[] = {212, 417, 112, 426};
    m = sizeof(blockSize) / sizeof(blockSize[0]);
    n = sizeof(processSize) / sizeof(processSize[0]);

    firstFit(blockSize, m, processSize, n);

    return 0 ;
}
```

Output:


C:\Users\kalya\OneDrive\Desktop\7.ipc.exe
Process No.	Process Size	Block no.
1	212	2
2	417	5
3	112	2
4	426	Not Allocated
Process exited after 0.03653 seconds with return value 0
Press any key to continue . . .