Q)Write a C program to simulate the following contiguous memory allocation techniques

a) Worst-fit

b) Best-fit

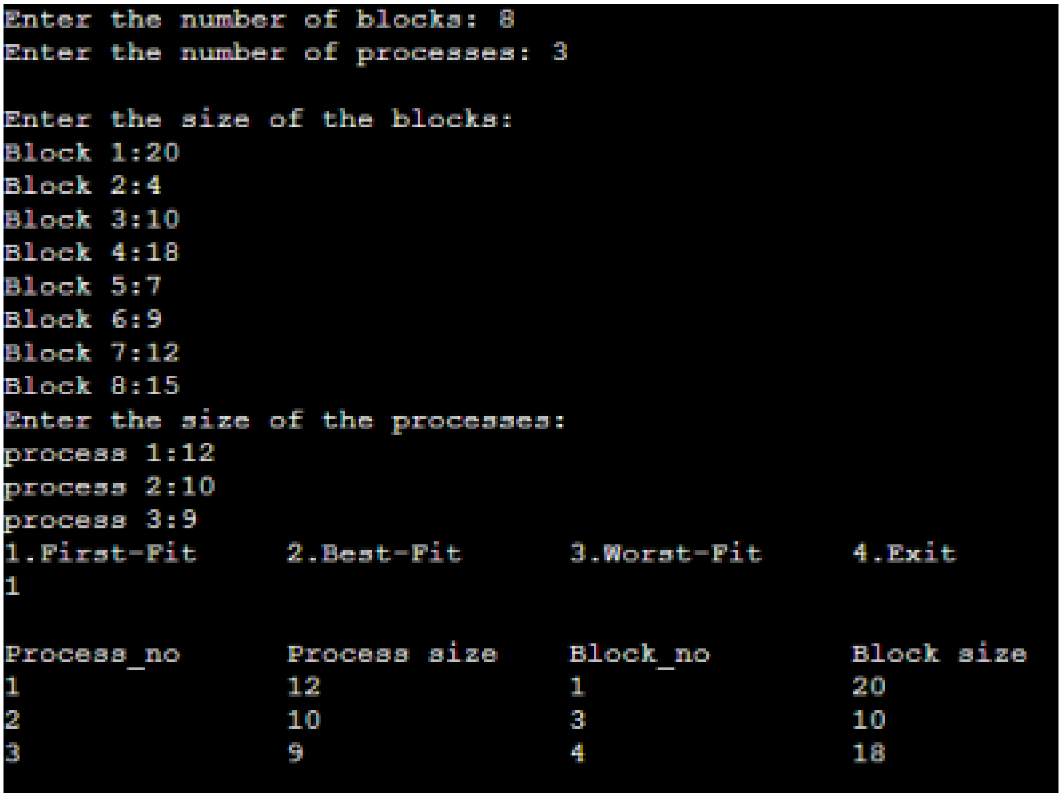
c) First-fit

```
#include <stdio.h>
#include<stdlib.h>
#define max 25

void readInput(int *nb, int *nf, int b[], int f[])
{
    int i;
    printf("Enter the number of blocks: ");
    scanf("%d", nb);
    printf("Enter the number of processes: ");
    scanf("%d", nf);
    printf("\nEnter the size of the blocks:\n");
    for (i = 1; i <= *nb; i++)
    {
        printf("Block %d:", i);
        scanf("%d", &b[i]);
    }
    printf("Enter the size of the processes:\n");
    for (i = 1; i <= *nf; i++)
    {
        printf("process %d:", i);
        scanf("%d", &f[i]);
    }
}

void bestFit(int nb, int nf, int b[], int f[], int bf[], int ff[])
{
    int i, j, temp, lowest = 999;
    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 (lowest>temp)
                    {
                        ff[i] = j;
                        lowest = temp;
                    }
                }
            }
        }
        bf[ff[i]] = 1;
        lowest = 999;
    }
}

void worstFit(int nb, int nf, int b[], int f[], int bf[], int ff[])
{
    int i, j, temp, lowest = 10000;
    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 (lowest == 10000 || temp > lowest)
                    {
                        ff[i] = j;
                        lowest = temp;
                    }
                }
            }
        }
        bf[ff[i]] = 1;
        lowest = 10000;
    }
}

void firstFit(int nb, int nf, int b[], int f[], int bf[], int ff[])
{
    int i, j, temp;
    for (i = 1; i <= nf; i++)
    {
        for (j = 1; j <= nb; j++)
        {
            if (bf[j] != 1)
            {
                temp = b[j] - f[i];
                if (temp >= 0)
                {
                    ff[i] = j;
                    break;
                }
            }
        }
        bf[ff[i]] = 1;
    }
}

void displayResults(int nf, int f[], int ff[], int b[])
{
    int i;

    printf("\nProcess_no\tProcess size\tBlock_no\tBlock size");
    for (i = 1; i <= nf; i++)
    {
        printf("\n%d\t\t%d\t\t%d\t\t%d\t", i, f[i], ff[i], b[ff[i]]);
    }
}

int main()
{
    int nb, nf, ch;
    int b[max], f[max], bf[max] ={0}, ff[max]={0}, frag[max]={0};
    readInput(&nb, &nf, b, f);
    printf("1.First-Fit\t2.Best-Fit\t3.Worst-Fit\t4.Exit\n");
    scanf("%d",&ch);
    switch(ch)
    {
        case 1: firstFit(nb, nf, b, f, bf, ff);
                break;
        case 2: bestFit(nb, nf, b, f, bf, ff);
                break;
        case 3: worstFit(nb, nf, b, f, bf, ff);
                break;
        case 4: exit(0);
```

```
                break;
        default: printf("Invalid choice\n");
                 break;
    }
    displayResults(nf, f, ff, b);
    return 0;
}
```

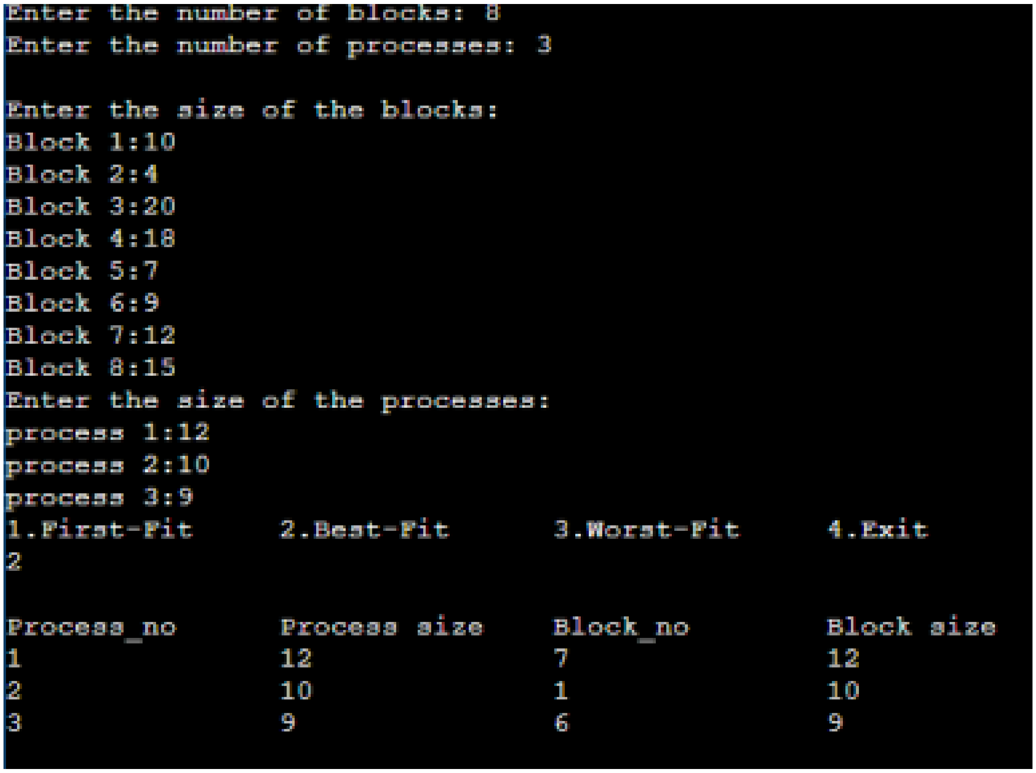
OUTPUT:

First-fit:



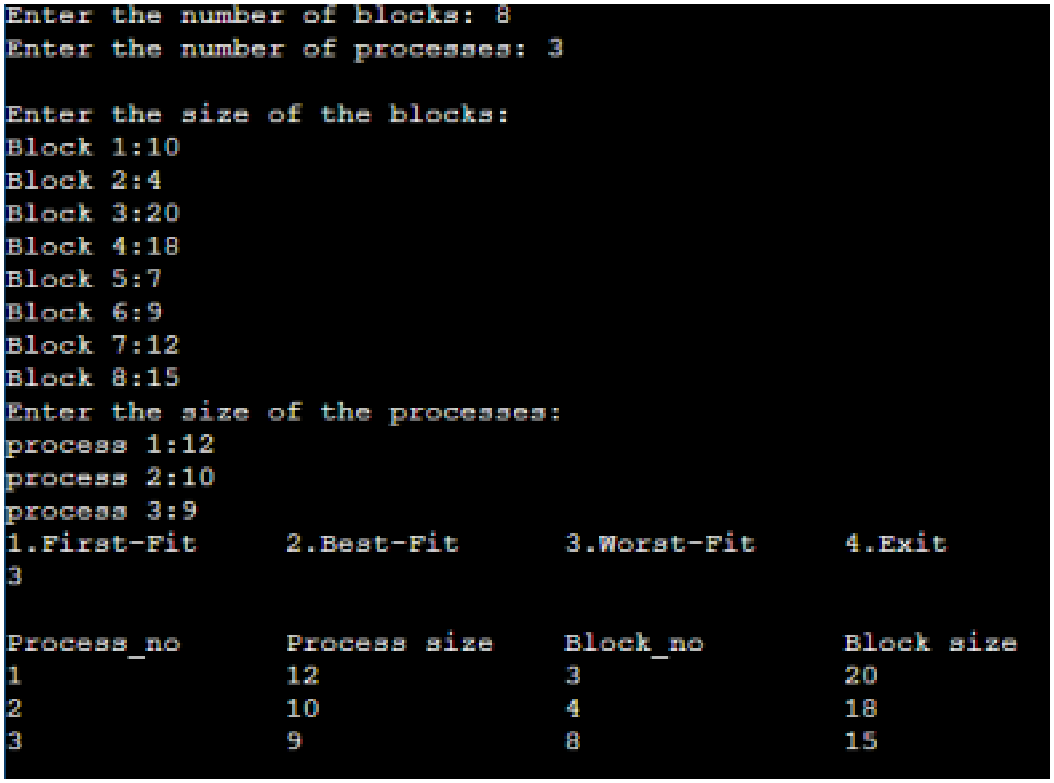
```
Enter the number of blocks: 8
Enter the number of processes: 3

Enter the size of the blocks:
Block 1:20
Block 2:4
Block 3:10
Block 4:18
Block 5:7
Block 6:9
Block 7:12
Block 8:15
Enter the size of the processes:
process 1:12
process 2:10
process 3:9
1.First-Fit     2.Best-Fit      3.Worst-Fit     4.Exit
1

Process_no      Process size    Block_no        Block size
1               12              1               20
2               10              3               10
3               9               4               18
```

Best_fit:



```
Enter the number of blocks: 8
Enter the number of processes: 3

Enter the size of the blocks:
Block 1:10
Block 2:4
Block 3:20
Block 4:18
Block 5:7
Block 6:9
Block 7:12
Block 8:15
Enter the size of the processes:
process 1:12
process 2:10
process 3:9
1.First-Fit       2.Best-Fit       3.Worst-Fit      4.Exit
2

Process_no        Process size     Block_no         Block size
1                 12               7                12
2                 10               1                10
3                 9                6                9
```

Worst_fit:



```
Enter the number of blocks: 8
Enter the number of processes: 3

Enter the size of the blocks:
Block 1:10
Block 2:4
Block 3:20
Block 4:18
Block 5:7
Block 6:9
Block 7:12
Block 8:15
Enter the size of the processes:
process 1:12
process 2:10
process 3:9
1.First-Fit       2.Best-Fit       3.Worst-Fit      4.Exit
3

Process_no        Process size     Block_no         Block size
1                 12               3                20
2                 10               4                18
3                 9                8                15
```