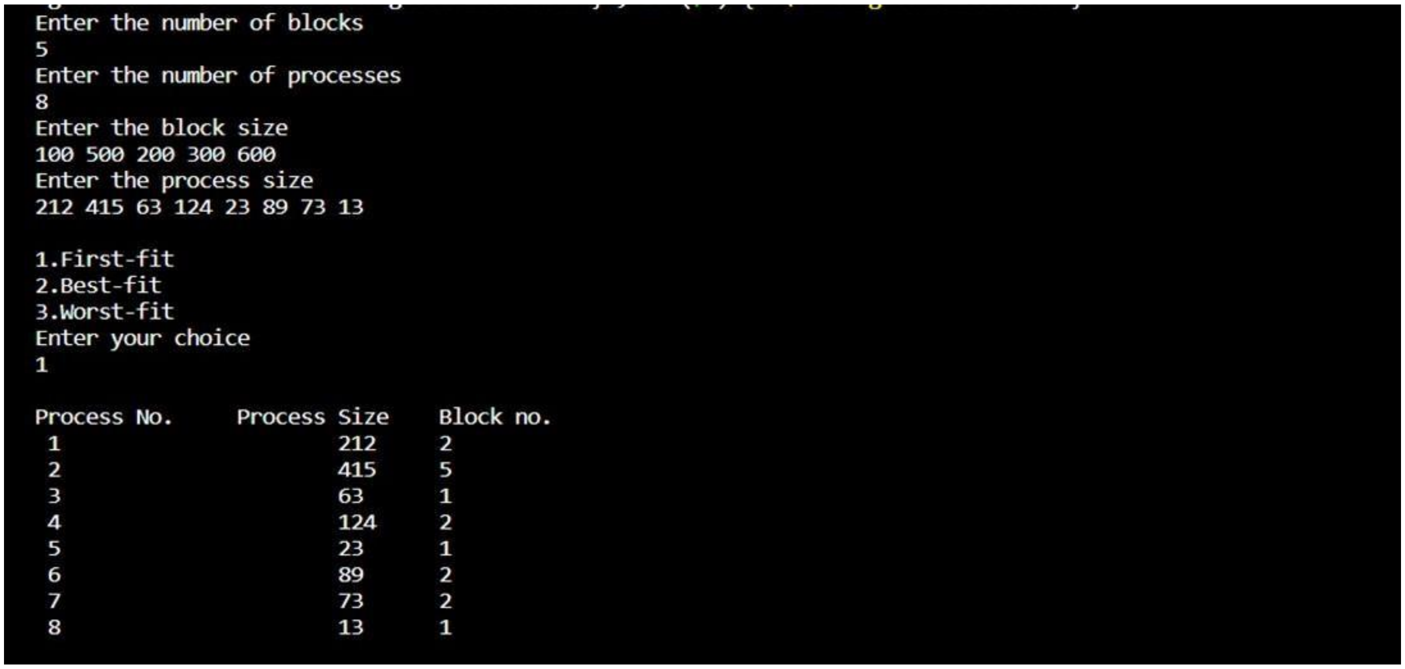
Program-6
Write a C program to simulate the following contiguous memory allocation techniques. a) Worst-fit b) Best-fit c) First-fit
Code:

```
#include
<stdio.h>
#define max 25
void firstFit(int b[], int nb, int
f[], int nf); void worstFit(int b[],
int nb, int f[], int nf); void
bestFit(int b[], int nb, int f[], int
nf);

int main()
{
   int b[max], f[max], nb, nf;

   printf("Memory Management Schemes\n");

   printf("\nEnter the number of blocks:");
scanf("%d", &nb);

   printf("Enter the number of files:");
scanf("%d", &nf);

   printf("\nEnter the size of the blocks:\n");
   for (int i = 1; i <= nb; i++)
   {
      printf("Block %d:", i);
      scanf("%d", &b[i]);
   }
   printf("\nEnter the size of the files:\n");
   for (int i = 1; i <= nf; i++)
   {
      printf("File %d:", i);
      scanf("%d", &f[i]);
   }
   printf("\nMemory Management Scheme - First Fit");
firstFit(b, nb, f, nf);

   printf("\n\nMemory Management Scheme - Worst Fit");
worstFit(b, nb, f, nf);

   printf("\n\nMemory Management Scheme -
Best Fit");     bestFit(b, nb, f, nf);     return 0;
```

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


printf("\nFile_no:\tFile_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]], frag[i]);}

void worstFit(int b[], int nb, int f[], int nf)
{
   int
bf[max] =
{0};     int
ff[max] =
{0};
   int frag[max], i, j, temp, highest = 0;

   for (i = 1; i <= nf; i++)
   {
      for (j = 1; j <= nb; j++)
      {
         if (bf[j] != 1)
         {
            temp = b[j] - f[i];
```

```
                if (temp >= 0 && highest < temp)
                {
             ff[i] = j;
                highest =
              temp;
                }
            }
        }
        frag[i] =
highest;
bf[ff[i]] = 1;
        highest = 0;
    }


printf("\nFile_no:\tFile_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]], frag[i]);     }
}

void bestFit(int b[], int nb, int f[], int nf)
{
    int
bf[max] =
{0};     int
ff[max] =
{0};
    int frag[max], 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 && lowest > temp)
                {
                    ff[i] = j;
                    lowest = temp;
                }
            }
        }
        frag[i] =
lowest;
bf[ff[i]] = 1;
        lowest = 10000;
    }


printf("\nFile_no:\tFile_size:\tBlock_no:\tBlock_size:\tFragment
");     for (i = 1; i <= nf && ff[i] != 0; i++)
    {
        printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", i, f[i], ff[i],
b[ff[i]], frag[i]);     }
}
```

Output:



```
Enter the number of blocks
5
Enter the number of processes
8
Enter the block size
100 500 200 300 600
Enter the process size
212 415 63 124 23 89 73 13

1.First-fit
2.Best-fit
3.Worst-fit
Enter your choice
1

Process No.     Process Size    Block no.
 1                      212     2
 2                      415     5
 3                      63      1
 4                      124     2
 5                      23      1
 6                      89      2
 7                      73      2
 8                      13      1
```