Program 6. Develop a C program to simulate the following contiguous memory allocation Techniques: a)Worst fit b)Best fit c)First fit.

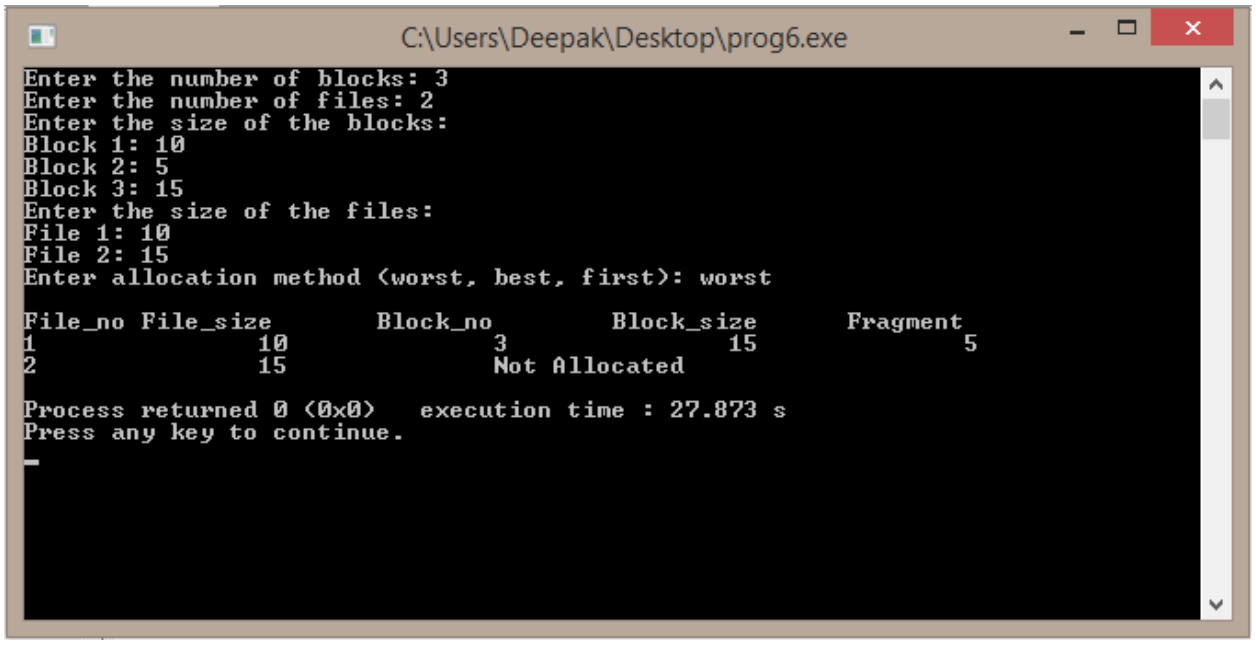
```
#include <stdio.h>
#include <string.h>  // For strcmp
#define MAX 25
void allocate(int b[], int f[], int nf, int nb, char* method);
int main() {
   int b[MAX], f[MAX], nb, nf, i;
   char method[10];

   printf("Enter the number of blocks: ");
   scanf("%d", &nb);
   printf("Enter the number of files: ");
   scanf("%d", &nf);
   printf("Enter the size of the blocks:\n");
   for (i = 0; i < nb; i++) {
      printf("Block %d: ", i + 1);
      scanf("%d", &b[i]);
   }
   printf("Enter the size of the files:\n");
   for (i = 0; i < nf; i++) {
      printf("File %d: ", i + 1);
      scanf("%d", &f[i]);
   }
   printf("Enter allocation method (worst, best, first): ");
   scanf("%s", method);
```

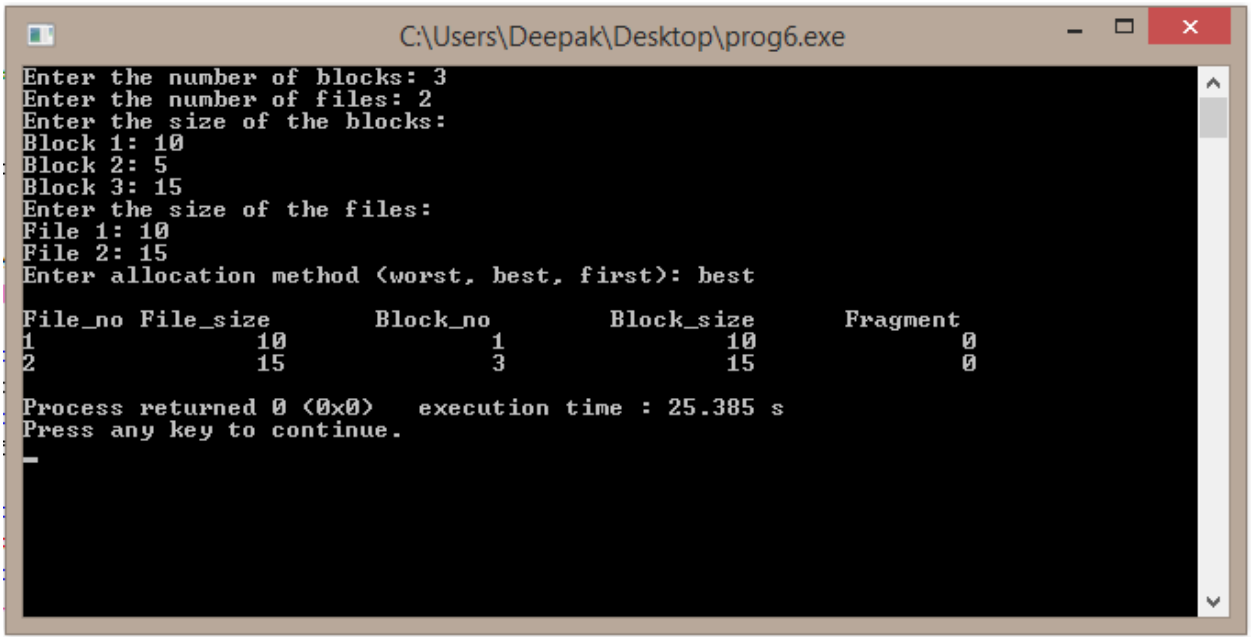
```
    allocate(b, f, nf, nb, method);
    return 0;
}

void allocate(int b[], int f[], int nf, int nb, char* method) {
    int bf[MAX], ff[MAX];
    int i, j, temp, highest, lowest, index;

    // Initialize allocation arrays
    for (i = 0; i < nf; i++) {
        ff[i] = -1; // -1 indicates that the file is not allocated
    }
    for (i = 0; i < nb; i++) {
        bf[i] = 0; // 0 indicates that the block is free
    }

    // Allocate files to blocks
    for (i = 0; i < nf; i++) {
        index = -1;

        if (strcmp(method, "worst") == 0) {
            highest = -1;
            for (j = 0; j < nb; j++) {
                if (bf[j] == 0 && b[j] >= f[i]) {
                    temp = b[j] - f[i];
                    if (temp > highest) {
                        highest = temp;
```

```
                index = j;
            }
        }
    }
} else if (strcmp(method, "best") == 0) {
    lowest = MAX;
    for (j = 0; j < nb; j++) {
        if (bf[j] == 0 && b[j] >= f[i]) {
            temp = b[j] - f[i];
            if (temp < lowest) {
                lowest = temp;
                index = j;
            }
        }
    }
} else if (strcmp(method, "first") == 0) {
    for (j = 0; j < nb; j++) {
        if (bf[j] == 0 && b[j] >= f[i]) {
            index = j;
            break;
        }
    }
} else {
    printf("Invalid allocation method.\n");
    return;
}
```

```
        if (index != -1) {
            ff[i] = index;
            bf[index] = 1; // Mark block as allocated
        }
    }
    // Print allocation results
    printf("\nFile_no\tFile_size\tBlock_no\tBlock_size\tFragment\n");
    for (i = 0; i < nf; i++) {
        printf("%d\t\t%d\t\t", i + 1, f[i]);
        if (ff[i] != -1) {
            printf("%d\t\t%d\t\t%d\n", ff[i] + 1, b[ff[i]], b[ff[i]] - f[i]);
        } else {
            printf("Not Allocated\n");
        }
    }
}
```

Output:




C:\Users\Deepak\Desktop\prog6.exe
Enter the number of blocks: 3
Enter the number of files: 2
Enter the size of the blocks:
Block 1: 10
Block 2: 5
Block 3: 15
Enter the size of the files:
File 1: 10
File 2: 15
Enter allocation method (worst, best, first): best
File_no File_size Block_no Block_size Fragment
1 10 1 10 0
2 15 3 15 0
Process returned 0 (0x0) execution time : 25.385 s
Press any key to continue.