WEEK 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>
#include <conio.h>
#define max 25

// Function for first fit algorithm
void firstfit()
{
   int frag[max], b[max], f[max], i, j, nb, nf, temp;
   static int bf[max], ff[max];

   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 (i = 1; i <= nb; i++)
   {
```

```
    printf("Block %d:", i);
    scanf("%d", &b[i]);
}
printf("Enter the size of the files:\n");
for (i = 1; i <= nf; i++)
{
    printf("File %d:", i);
    scanf("%d", &f[i]);
}
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;
            }
        }
    }
    frag[i] = temp;
    bf[ff[i]] = 1;
}
```

```
    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]);
}

// Function for best fit algorithm
void bestfit()
{
    int frag[max], b[max], f[max], i, j, nb, nf, temp, lowest = 10000;
    static int bf[max], ff[max];

    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 (i = 1; i <= nb; i++)
    {
        printf("Block %d:", i);
        scanf("%d", &b[i]);
    }
    printf("Enter the size of the files:\n");
    for (i = 1; i <= nf; i++)
    {
        printf("File %d:", i);
        scanf("%d", &f[i]);
```

```
    }
    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;
                    }
            }
        }
        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]);
}

// Function for worst fit algorithm
```

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

   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 (i = 1; i <= nb; i++)
   {
      printf("Block %d:", i);
      scanf("%d", &b[i]);
   }
   printf("Enter the size of the files:\n");
   for (i = 1; i <= nf; i++)
   {
      printf("File %d:", i);
      scanf("%d", &f[i]);
   }
   for (i = 1; i <= nf; i++)
   {
      for (j = 1; j <= nb; j++)
      {
         if (bf[j] != 1) //if bf[j] is not allocated
```

```
        {
            temp = b[j] - f[i];
            if (temp >= 0)
                if (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 main()
{
    int c;
    while (1)
    {
        printf("\n1. First Fit\n2. Best Fit\n3. Worst Fit\n4. Exit");
        printf("\nEnter choice:");
```

```
        scanf("%d", &c);
        switch (c)
        {
        case 1:
           firstfit();
           break;
        case 2:
           bestfit();
           break;
        case 3:
           worstfit();
           break;
        case 4:
           exit(0);
        default:
           printf("Invalid choice");
        }
    }
}
```

Output:

```
1. First Fit
2. Best Fit
3. Worst Fit
4. Exit
Enter choice:1

Enter the number of blocks:8
Enter the number of files: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 files:
File 1:12
File 2:10
File 3:9

File_no:        File_size:      Block_no:       Block_size:     Fragment
1               12              3               20              8
2               10              1               10              0
3               9               4               18              9
```

```
1. First Fit
2. Best Fit
3. Worst Fit
4. Exit
Enter choice:2

Enter the number of blocks:8
Enter the number of files: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 files:
File 1:12
File 2:10
File 3:9

File No File Size       Block No        Block Size      Fragment
1               12              7               12              0
2               10              1               10              0
3               9               6               9               0
```

```
1. First Fit
2. Best Fit
3. Worst Fit
4. Exit
Enter choice:3

Enter the number of blocks:8
Enter the number of files: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 files:
File 1:12
File 2:10
File 3:9

File_no:        File_size:      Block_no:       Block_size:     Fragment
1               12              3               20              8
2               10              4               18              8
3               9               8               15              6
1. First Fit
2. Best Fit
3. Worst Fit
4. Exit
Enter choice:4
```