WEEK 7 OS

1BM21CS255

MEMORY MANAGEMENT ALGORITHMS

CODE

```
#include <stdio.h>
#include<stdlib.h>
#define max 25
void readInput(int *nb, int *nf, int b[], int f[]);
void bestFit(int nb, int nf, int b[], int f[], int bf[], int ff[], int frag[]);
void worstFit(int nb, int nf, int b[], int f[], int bf[], int ff[], int frag[]);
void firstFit(int nb, int nf, int b[], int f[], int bf[], int ff[], int frag[]);
void displayResults(int nf, int f[], int ff[], int b[], int frag[]);
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.Best Fit 2.Worst Fit 3.First Fit 4. Exit\n");
   scanf("%d",&ch);
   switch(ch)
   {
      case 1: bestFit(nb, nf, b, f, bf, ff, frag);
            break;
      case 2: worstFit(nb, nf, b, f, bf, ff, frag);
            break;
      case 3: firstFit(nb, nf, b, f, bf, ff, frag);
```

```
            break;
        case 4: exit(0);
            break;
        default: printf("Inavlid choice\n");
             break;
    }
    displayResults(nf, f, ff, b, frag);
    return 0;
}

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 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]);
    }
}

void bestFit(int nb, int nf, int b[], int f[], int bf[], int ff[], int frag[])
{
    int i, j, temp, lowest = 10000;

    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 (lowest>temp)
                    {
                        ff[i] = j;
                        lowest = temp;
                    }
                }
            }
        }
        frag[i] = lowest;
        bf[ff[i]] = 1;
        lowest = 10000;
```

```
    }
}

void worstFit(int nb, int nf, int b[], int f[], int bf[], int ff[], int frag[])
{
    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;
                    }
                }
            }
        }
        frag[i] = lowest;
        bf[ff[i]] = 1;
        lowest = 10000;
    }
}
```

```
void firstFit(int nb, int nf, int b[], int f[], int bf[], int ff[], int frag[])
{
   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;
            }
         }
      }
      frag[i] = temp;
      bf[ff[i]] = 1;
   }
}

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

   printf("\nFile_no\t\tFile_size\t Block_size\t");
```

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

OUTPUTS

FIRST FIT OUTPUT

```
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
1.Best Fit 2.Worst Fit 3.First Fit 4. Exit
3

File_no         File_size        Block_size
1               12              20
2               10              10
3               9               18

...Program finished with exit code 0
Press ENTER to exit console.
```

WORST FIT OUTPUT

```
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
1.Best Fit 2.Worst Fit 3.First Fit 4. Exit
2

File_no         File_size        Block_size
1               12              20
2               10              18
3               9               15

...Program finished with exit code 0
Press ENTER to exit console.
```

BEST FIT OUTPUT