First Fit: Allocates the first block that is large enough for a file.

- `MAX`: Maximum number of blocks/files.
- `b[]`: Array holding **block sizes**.
- `f[]`: Array holding **file sizes**.
- `frag[]`: Holds **internal fragmentation** (unused space in block).
- `bf[]`: Marks if block is **used (1)** or **free (0)**.
- `ff[]`: Maps each **file to a block number**.

Best Fit: To implement the **Best Fit Memory Allocation Strategy**, where:

Each file is allocated to the *smallest block* that is *just large enough* to hold it.
This helps minimize **wasted space (fragmentation)**.

- `frag[MAX]`: Stores leftover space (fragment) after allocation
- `bf[MAX]`: Tracks which blocks are already **used**
- `ff[MAX]`: Stores which block each file is allocated to

Task 10

Task 10a: To write C program to simulate First Fit Algorithm of Memory Management

Program:

```
#include <stdio.h>
#define MAX 25
int main()
{
   int frag[MAX], b[MAX], f[MAX], i, j, nb, nf, temp;
   static int bf[MAX], ff[MAX];

   printf("Memory Management Scheme - First Fit");

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

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

   printf("Enter size of each block:\n");
   for(i = 0; i < nb; i++) {
      printf("Block %d: ", i+1);
      scanf("%d", &b[i]);
   }
```

```
printf("Enter size of each file:\n");
for(i = 0; i < nf; i++) {
    printf("File %d: ", i+1);
    scanf("%d", &f[i]);
}

for(i = 0; i < nf; i++) {
    for(j = 0; j < nb; j++) {
        if(bf[j] != 1) { // if block is not allocated
            temp = b[j] - f[i];
            if(temp >= 0) {
                ff[i] = j;
                frag[i] = temp;
                bf[j] = 1;
                break;
            }
        }
    }
    if(j == nb) { // no block found
        ff[i] = -1;
        frag[i] = -1;
    }
    }

    printf("\nFile No\tFile Size\tBlock No\tBlock Size\tFragment");
    for(i = 0; i < nf; i++) {
        if(ff[i] != -1)
            printf("\n%d\t%d\t\t%d\t\t%d\t\t%d", i+1, f[i], ff[i]+1, b[ff[i]], frag[i]);
        else
            printf("\n%d\t%d\t\tNot Allocated", i+1, f[i]);
    }

    return 0;
}
```

Output:

```
Memory Management Scheme - First Fit
Enter number of blocks: 3
Enter number of files: 3
Enter size of each block:
Block 1: 200
Block 2: 300
Block 3: 100
Enter size of each file:
File 1: 280
File 2: 80
```

File 3: 50

File No	**File Size**	**Block No**	**Block Size**	**Fragment**
1	**280**	**2**	**300**	**20**
2	**80**	**1**	**200**	**120**
3	**50**	**3**	**100**	**50**

Task 10b: To write C program to simulate Best Fit Algorithm of Memory Management

Program:

```c
#include <stdio.h>
#define MAX 25

int main() {
    int frag[MAX], b[MAX], f[MAX], i, j, nb, nf, temp, lowest;
    static int bf[MAX], ff[MAX];

    printf("Memory Management Scheme - Best Fit");

    // Input number of memory blocks and files
    printf("\nEnter the number of blocks: ");
    scanf("%d", &nb);
    printf("Enter the number of files: ");
    scanf("%d", &nf);

    // Input block sizes
    printf("\nEnter the size of the blocks:\n");
    for (i = 0; i < nb; i++) {
        printf("Block %d: ", i + 1);
        scanf("%d", &b[i]);
    }
```

```c
// Input file sizes
printf("\nEnter the size of the files:\n");
for (i = 0; i < nf; i++) {
    printf("File %d: ", i + 1);
    scanf("%d", &f[i]);
}

// Best Fit allocation
for (i = 0; i < nf; i++) {
    lowest = 10000;
    ff[i] = -1;

    for (j = 0; j < nb; j++) {
        temp = b[j] - f[i];
        if (temp >= 0 && bf[j] == 0) {
            if (temp < lowest) {
                ff[i] = j;
                lowest = temp;
            }
        }
    }
```

```
        if (ff[i] != -1) {
            frag[i] = b[ff[i]] - f[i];
            bf[ff[i]] = 1;
        }
    }

    // Output the allocation result
    printf("\nFile No\tFile Size\tBlock No\tBlock Size\tFragment");
    for (i = 0; i < nf; i++) {
        if (ff[i] != -1)
            printf("\n%d\t%d\t\t%d\t\t%d\t\t%d", i + 1, f[i], ff[i] + 1, b[ff[i]], frag[i]);
        else
            printf("\n%d\t%d\t\tNot Allocated", i + 1, f[i]);
    }

    return 0;
}
```

Output:

```
Memory Management Scheme - Best Fit
Enter the number of blocks: 4
Enter the number of files: 3
Enter the size of the blocks:

Block 1: 100

Block 2: 500

Block 3: 200

Block 4: 300


Enter the size of the files:

File 1: 212

File 2: 417

File 3: 112


File No    File Size      Block No       Block Size     Fragment

1            212              4            300             88

2             417              2           300             83

3             112              3           200              88
```