ABIRAMI K

230701008 CSE-A

EXP NO: 10

EXP 10A BEST FIT

PROGRAM

```c
#include <stdio.h>

#define MAX 100

int main() {
    int blockSize[MAX], processSize[MAX];
    int blockCount, processCount;
    int allocation[MAX]; // To store the block index assigned to each process

    printf("Enter the number of memory blocks: ");
    scanf("%d", &blockCount);

    printf("Enter the sizes of the %d memory blocks:\n", blockCount);
    for (int i = 0; i < blockCount; i++) {
        scanf("%d", &blockSize[i]);
    }

    printf("Enter the number of processes: ");
    scanf("%d", &processCount);

    printf("Enter the sizes of the %d processes:\n", processCount);
    for (int i = 0; i < processCount; i++) {
        scanf("%d", &processSize[i]);
    }

    // Initialize all allocations to -1 (not allocated)
    for (int i = 0; i < processCount; i++) {
        allocation[i] = -1;
    }

    // Best Fit allocation
    for (int i = 0; i < processCount; i++) {
        int bestIdx = -1;
        for (int j = 0; j < blockCount; j++) {
            if (blockSize[j] >= processSize[i]) {
                if (bestIdx == -1 || blockSize[j] < blockSize[bestIdx]) {
                    bestIdx = j;
                }
            }
        }
```

```c
        // If a suitable block is found
        if (bestIdx != -1) {
            allocation[i] = bestIdx;
            blockSize[bestIdx] -= processSize[i]; // Reduce the available block
        }
    }

    // Output results
    printf("\nProcess No.\tProcess Size\tBlock No.\n");
    for (int i = 0; i < processCount; i++) {
        printf("%d\t\t%d\t\t", i + 1, processSize[i]);
        if (allocation[i] != -1)
            printf("%d\n", allocation[i] + 1); // +1 for 1-based indexing
        else
            printf("Not Allocated\n");
    }

    return 0;
}
```

OUTPUT

```
Enter the number of memory blocks: 5
Enter the sizes of the 5 memory blocks:
100 500 200 300 600
Enter the number of processes: 4
Enter the sizes of the 4 processes:
212
417
112
426

Process No.     Process Size    Block No.
1               212             4
2               417             2
3               112             3
4               426             5
```

EXP NO: 10B BEST FIT

PROGRAM

```c
#include <stdio.h>

#define max 25

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

    // Step 3: Get number of blocks and files
    printf("Enter the number of blocks: ");
    scanf("%d", &nb);

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

    printf("Enter the size of each block:\n");
    for (i = 0; i < nb; i++) {
        printf("Block %d: ", i + 1);
        scanf("%d", &b[i]);
        bf[i] = 0; // initially all blocks are free
    }

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

    // Step 4 & 5: First Fit Allocation
    for (i = 0; i < nf; i++) {
        for (j = 0; j < nb; j++) {
            if (bf[j] == 0 && b[j] >= f[i]) { // block is free and large enou
                ff[i] = j;                    // allocate block j to file i
                bf[j] = 1;                    // mark block as filled
                frag[i] = b[j] - f[i];        // calculate fragmentation
                break;
            }
        }
        if (j == nb) {
            ff[i] = -1; // no suitable block found
            frag[i] = -1;
        }
    }
```

```
    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]], frag[i]);
        else
            printf("Not Allocated\t-\t\t-\n");
    }

    return 0;
}
```

OUTPUT

```
Enter the number of blocks: 3
Enter the number of files: 2
Enter the size of each block:
Block 1: 100
Block 2: 500
Block 3: 200
Enter the size of each file:
File 1: 212
File 2: 417

File No.        File Size       Block No.       Block Size      Fragment
1               212             2               500             288
2               417             Not Allocated   -               -
```