10a.

CODE-

```python
def best_fit(block_size, process_size):
    m = len(block_size)  # Number of blocks
    n = len(process_size)  # Number of processes

    allocation = [-1] * n  # Stores block index assigned to each process

    # Pick each process and find the best fit block
    for i in range(n):
        best_index = -1
        for j in range(m):
            if block_size[j] >= process_size[i]:
                if best_index == -1 or block_size[j] < block_size[best_index]:
                    best_index = j

        # If found, allocate the block
        if best_index != -1:
            allocation[i] = best_index
            block_size[best_index] -= process_size[i]

    # Display the result
    print("Process No.\tProcess Size\tBlock no.")
    for i in range(n):
        print(f"{i + 1}\t\t{process_size[i]}\t\t{allocation[i] + 1 if
            allocation[i] != -1 else 'Not Allocated'}")


# Example Data
block_size = [100, 500, 200, 300, 600]
process_size = [212, 417, 112, 426]

best_fit(block_size, process_size)
```

OUTPUT-

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

10b.

CODE-

cse16@localhost:~

```
#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("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 = 0; i < nb; i++) {
        printf("Block %d: ", i + 1);
        scanf("%d", &b[i]);
        bf[i] = 0; // Initially mark block as free
    }

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

    // First Fit Allocation
    for(i = 0; i < nf; i++) {
        for(j = 0; j < nb; j++) {
            if(bf[j] == 0 && b[j] >= f[i]) {
                ff[i] = j; // allocate block j to file i
                frag[i] = b[j] - f[i];
                bf[j] = 1; // mark block as allocated
                break;
            }
        }
    }

    // Displaying Output
    printf("\nFile_no:\tFile_size:\tBlock_no:\tBlock_size:\tFragment\n");
    for(i = 0; i < nf; i++) {
        printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n",
            i + 1,
            f[i],
            ff[i] + 1,
            b[ff[i]],
            frag[i]
        );
    }

    return 0;
}
```

OUTPUT-

```
Enter the number of blocks: 4
Enter the number of files: 3

Enter the size of the blocks:-
Block 1: 5
Block 2: 8
Block 3: 4
Block 4: 10

Enter the size of the files:-
File 1: 1
File 2: 4
File 3: 7

File_no:        File_size:      Block_no:       Block_size:     Fragment
1               1               1               5               4
2               4               2               8               4
3               7               4               10              3
```