Ex. No.: 10a)
Date: 19.04.2025

BEST FIT

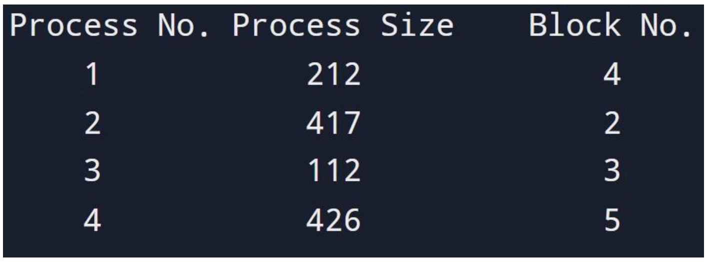
Aim:

To implement Best Fit memory allocation technique using Python.

Program:

```
def best_fit(block_size, process_size):
    n = len(block_size)
    m = len(process_size)
    allocation = [-1] * m

    for i in range(m):
        best_idx = -1
        for j in
        range(n):
            if block_size[j] >= process_size[i]:
                if best_idx == -1 or block_size[j] < block_size[best_idx]:
                    best_idx = j
        if best_idx != -1:
            allocation[i] = best_idx + 1
            block_size[best_idx] -= process_size[i]

    print("Process No.\tProcess Size\tBlock No.")
    for i in range(m):
        print(f"{i+1}\t\t{process_size[i]}\t\t", end="")
        if allocation[i] != -1:
            print(f"{allocation[i]}")
        else:
            print("Not Allocated")

# Sample input
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
```

Result:

The program for Best Fit memory allocation technique was executed successfully and the output was verified.

Ex. No.: 10b)
Date: 19.04.2025

FIRST FIT

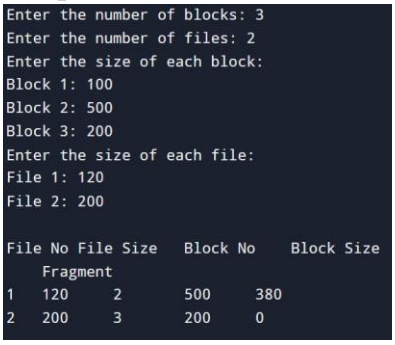
Aim:

To write a C program for implementation memory allocation methods for fixed partition using first fit.

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("Enter the number of blocks: ");
   scanf("%d", &nb);
   printf("Enter the number of files: ");
   scanf("%d", &nf);
   printf("Enter the size of the blocks:\n");
   for (i = 0; i < nb; i++)
      scanf("%d", &b[i]);
   printf("Enter the size of the files:\n");
   for (i = 0; i < nf; i++)
      scanf("%d", &f[i]);
   for (i = 0; i < nf; i++) {
      for (j = 0; j < nb; j++) {
         if (bf[j] != 1 && b[j] >= f[i]) {
            ff[i] = j;
            bf[j] = 1;
            frag[i] = b[j] - f[i];
            break;
         }
      }
   }
   printf("\nFile No\tFile Size\tBlock No\tBlock Size\tFragment\n");
   for (i = 0; i < nf; i++)
      printf("%d\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: 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: 120
File 2: 200

File No File Size   Block No    Block Size
    Fragment
1   120     2       500     380
2   200     3       200     0
```

Result:

Thus, the program for First Fit memory allocation technique was executed successfully and the output was verified.