Ex. No.: 10a) **NAME: T.R.DIVYASREE**

ROLL NO: 230701083

BEST FIT

Aim:

To implement Best Fit memory allocation technique using Python.

Program Code:

```
#include <stdio.h>


#define MAX 25

int main() {
      int blockSize[MAX], processSize[MAX];
      int allocation[MAX], blockCount, processCount;

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

      printf("Enter the size of each memory
   block:\n"); for (int i = 0; i < blockCount; i+
   +) {
            printf("Block %d: ", i + 1);
            scanf("%d",  &blockSize[i]);
      }
      printf("Enter the number of processes: ");

   scanf("%d", &processCount);

      printf("Enter the size of each
   process:\n"); for (int i = 0; i <
   processCount; i++) {
            printf("Process %d:
      ", i + 1); scanf("%d",
      &processSize[i]);
            allocation[i] = -1; // initialize as not allocated
      }

         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;
                  }
            }

               if (bestIdx != -1)
```

{ allocation[i] =

```
bestIdx;
            blockSize[bestIdx] -= processSize[i];
    }
```

```
    }

    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); // block
    numbers start from 1 else
          printf("Not Allocated\n");
    }

    return 0;
}
```

```
Enter the number of memory blocks: 5
Enter the size of each memory block:
Block 1: 100
Block 2: 2000
Block 3: 300
Block 4: 400
Block 5: 500
Enter the number of processes: 4
Enter the size of each process:
Process 1: 212
Process 2: 417
Process 3: 112
Process 4: 426

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

OUTPUT :

Ex. No.: 10b) **Roll No.:230701091**

FIRST FIT

Aim:

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

Program Code :

```
#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;

    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; //
    block free
    }

    printf("Enter the 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] == 0 && b[j] >= f[i]) { // first free
        block large enough ff[i] = j;
              frag[i] = b[j] - f[i];
              bf[j] = 1; // mark block
        as allocated 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++) {
            if (ff[i] != -1) {
                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]);
            } else {
                printf("%d\t\t%d\t\tNot Allocated\n", i + 1, f[i]);
            }
        }
        return 0;
}
```

```
Enter the number of blocks: 5
Enter the number of files: 4
Enter the size of each block:
Block 1: 100
Block 2: 500
Block 3: 200
Block 4: 300
Block 5: 600
Enter the size of each file:
File 1: 212
File 2: 417
File 3: 112
File 4: 426

File No.        File Size       Block No.       Block Size      Fragment
1               212             2               500             288
2               417             5               600             183
3               112             3               200             88
4               426             Not Allocated
```

OUTPUT :