Ex. No.: 10a **BEST FIT**

Date: 16.4.2025

Aim:

To implement the Best Fit memory allocation technique .

Algorithm:

1. Input memory blocks and processes with their sizes.
2. Initialize all memory blocks as free.
3. For each process, find the smallest memory block that can accommodate it.
4. If such a block is found, allocate it to the process.
5. If no suitable block is found, leave the process unallocated.

Program Code :

```
#include <stdio.h>

#define MAX_BLOCKS 10
#define MAX_PROCESSES 10

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

  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("\nEnter 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;
}

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 + 1;
        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]);
        else
            printf("Not Allocated\n");
    }

    return 0;
}
```

Sample Output:

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

Result:

Thus, the Best Fit memory allocation technique was successfully implemented .

Ex. No.: 10b **FIRST FIT**

Date: 16.4.2025

Aim:

To write a C program for implementation of memory allocation methods for fixed partition using First Fit.

Algorithm:

1. Define the maximum limit as #define max 25.
2. Declare variables: frag[max], b[max], f[max], i, j, nb, nf, temp, bf[max], ff[max].
3. Input the number of blocks (nb) and files (nf).
4. Input the size of each block and file using loops.
5. For each file, search for the first block that is free and large enough to accommodate it.
6. If found, allocate that block to the file and calculate internal fragmentation.
7. Mark the block as used.
8. Print the allocated block and fragmentation details.

Program Code

```
#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 number of blocks: ");
    scanf("%d", &nb);

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

    printf("\nEnter size of each block:\n");
    for (i = 0; i < nb; i++) {
        printf("Block %d: ", i + 1);
        scanf("%d", &b[i]);
        bf[i] = 0;
    }
    printf("\nEnter 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]) {
                ff[i] = j;
                frag[i] = b[j] - f[i];
                bf[j] = 1;
                break;
            }
        }
        if (j == nb) {
            ff[i] = -1;
            frag[i] = 0;
        }
    }

    printf("\nFile No\tFile Size\tBlock No\tBlock Size\tFragment\n");
    for (i = 0; i < nf; i++) {
        printf("%d\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("0\t\t0\t\t0\n");
    }

    return 0;
}
```

Sample Output:

```
Enter number of blocks: 5
Enter number of files: 4


Enter size of each block:
Block 1: 100
Block 2: 500
Block 3: 200
Block 4: 300
Block 5: 600
```

```
Enter 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        0      0                  0          <-- Not allocated
```

Result:

Thus, the First Fit memory allocation technique for fixed partitioning was implemented successfully in C