Ex. No. : 10a **Date : 11.04.2025**

Register No. : 230701385 **Name : VISHWAK S**

BEST FIT

Aim:

To implement the Best Fit memory allocation technique.

Algorithm:

1. Input memory blocks and processes with sizes

2. Initialize all memory blocks as free.

3. Start by picking each process and find the minimum block size that ca assigned to current process

4. If found then assign it to the current process.

5. If not found then leave that process and keep checking the further processes.

Program:

D: > OS > C Best_fit.c > ...

```c
#include <stdio.h>
int main()
{
    int n, m;
    printf("Enter number of blocks : ");
    scanf("%d", &n);
    printf("\nEnter number of processes: ");
    scanf("%d", &m);
    int blocks[n];
    int process[m];
    int allocation[m];
    for (int i = 0; i < m; i++)
    {
        allocation[i] = -1;
    }
    for (int i = 0; i < n; i++)
    {
        printf("\nEnter block %d size : ", i + 1);
        scanf("%d", &blocks[i]);
    }
    for (int i = 0; i < m; i++)
    {
        printf("\nEnter process %d size : ", i + 1);
        scanf("%d", &process[i]);
    }
    int best_index;
    for (int i = 0; i < m; i++)
    {
        best_index = -1;
        for (int j = 0; j < n; j++)
        {
            if (blocks[j] >= process[i])
            {
                if (best_index == -1 || blocks[j] < blocks[best_index])
                {
                    best_index = j;
                }
            }
        }
```

```
        if (best_index != -1)
        {
            allocation[i] = best_index;
            blocks[best_index] -= process[i];
        }
    }

    printf("\nProcess No.           Process Size            Block No.");
    for (int i = 0; i < m; i++)
    {
        if (allocation[i] != -1)
        {
            printf("\n%d\t\t\t%d\t\t\t%d", i + 1, process[i], allocation[i] + 1);
        }
        else
        {
            printf("\n%d\t\t\t%d\t\t\tNot Allocated", i + 1, process[i]);
        }
    }
}
```

Output:

```
PS D:\OS> cd "d:\OS\" ; if ($?) { gcc Best_fit.c -o Best_fit.exe } ; if ($?) { ./Best_fit.exe }
Enter number of blocks : 4

Enter number of processes: 3

Enter block 1 size : 100

Enter block 2 size : 500

Enter block 3 size : 150

Enter block 4 size : 300

Enter process 1 size : 99

Enter process 2 size : 211

Enter process 3 size : 300

Process No.           Process Size            Block No.
1                       99                      1
2                       211                     4
3                       300                     2
PS D:\OS>
```

Result:

Hence a program to implement Best Fit memory allocation technique has been executed successfully.

Ex. No. : 10b **Date : 11.04.2025**

Register No. : 230701385 **Name : VISHWAK S**

FIRST FIT

Aim:

To implement the First Fit memory allocation technique.

Algorithm:

1. Define the max as 25.
2. Declare the variable frag[max],b[max],f[max],i,j,nb,nf,temp, highest=0, bf[max],ff[max].
3. Get the number of blocks,files,size of the blocks using a for loop.
4. In for loop check bf[j]!=1, if so temp=b[j]-f[i]
5. Check highest

Program:

D: > OS > C First_fit.c > ...

```c
#include <stdio.h>
int main()
{
    int n, m;
    printf("Enter number of blocks : ");
    scanf("%d", &n);
    printf("\nEnter number of processes: ");
    scanf("%d", &m);
    int blocks[n];
    int process[m];
    int allocation[m];
    for (int i = 0; i < m; i++)
    {
        allocation[i] = -1;
    }
    int occupied[n];
    for (int i = 0; i < n; i++)
    {
        occupied[i] = 0;
    }
    for (int i = 0; i < n; i++)
    {
        printf("\nEnter block %d size : ", i + 1);
        scanf("%d", &blocks[i]);
    }
    for (int i = 0; i < m; i++)
    {
        printf("\nEnter process %d size : ", i + 1);
        scanf("%d", &process[i]);
    }
```

```
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (!occupied[j] && blocks[j] >= process[i])
            {
                allocation[i] = j;
                occupied[j] = 1;
                blocks[j] -= process[i];
                printf("%d", blocks[j]);
                break;
            }
        }
    }
    printf("\nProcess No.           Process Size            Block No.");
    for (int i = 0; i < m; i++)
    {
        if (allocation[i] != -1)
        {
            printf("\n%d\t\t\t%d\t\t\t%d", i + 1, process[i], allocation[i] + 1);
        }
        else
        {
            printf("\n%d\t\t\t%d\t\t\tNot Allocated", i + 1, process[i]);
        }
    }
}
```

Output:

```
PS D:\OS> cd "d:\OS\" ; if ($?) { gcc First_fit.c -o First_fit.exe } ; if ($?) { ./First_fit.exe }
Enter number of blocks : 4

Enter number of processes: 3

Enter block 1 size : 100

Enter block 2 size : 500

Enter block 3 size : 150

Enter block 4 size : 300

Enter process 1 size : 99

Enter process 2 size : 211

Enter process 3 size : 300
12890
Process No.           Process Size            Block No.
1                       99                      1
2                       211                     2
3                       300                     4
PS D:\OS>
```

Result:

Hence a program to implement First Fit memory allocation technique has been executed successfully.