Ex. no: 10a

Name : Ashish P Shaji

Roll NO : 230701041

BEST FIT

Aim:

To implement Best Fit memory allocation technique using Python.

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 can be 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 Code:

```
def best_fit(memory_blocks, processes):
    allocation = [-1] * len(processes)  # Initialize allocation for each process

    for i in range(len(processes)):
        best_index = -1
        for j in range(len(memory_blocks)):
            if memory_blocks[j] >= processes[i]:
                if best_index == -1 or memory_blocks[j] < memory_blocks[best_index]:
                    best_index = j

        # If a suitable block is found
        if best_index != -1:
            allocation[i] = best_index + 1  # Allocate block (using 1-based index)
            memory_blocks[best_index] -= processes[i]  # Update available size of block

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


# Input: Memory blocks and process sizes
memory_blocks = [100, 500, 200, 300, 600]
processes = [212, 417, 112, 426]

best_fit(memory_blocks, processes)
~
~
```

OUTPUT :

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

Ex. no: 10b

Name : Ashish P Shaji

Roll NO : 230701041

FIRST FIT

Aim:

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

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 for loop.

4: In for loop check bf[j]!=1, if so temp=b[j]-f[i]

5: Check highest

Program Code:

```c
#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("\nEnter 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 = 1; i <= nb; i++)
    {
        printf("Block %d: ", i);
        scanf("%d", &b[i]);
    }

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

    for (i = 1; i <= nf; i++)
    {
        for (j = 1; j <= nb; j++)
        {
            if (bf[j] != 1)
            {
                temp = b[j] - f[i];
                if (temp >= 0)
                {
                    ff[i] = j;
                    frag[i] = temp;
                    bf[j] = 1;
                    break;
                }
            }
        }
    }

    printf("\nFile_no:\tFile_size:\tBlock_no:\tBlock_size:\tFragment");
    for (i = 1; i <= nf; i++)
    {
        printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",
               i, f[i], ff[i], 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
```