Ex. No.: 10a)
Date: 09-04-25

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:

```
# include <stdio.h>

void bestFit ( int blocks[], int m, int porocessess[], int n) {
        int allocation[n];
        for ( int i=0; i<n; i++){ allocation [i] = -1; }
        for ( int i=0; i<n; i++){
                int bestIdx = -1;
                for (int j=0; j<m; j++){
                    if (blocks [j] >= porocesses[i] && allocation [i]==-1){
                      if ( best Idx ==-1)|| blocks [bestIdx]>blocks[j]){
                              bestIdx=j;
                      }
                }
        }

      if (bestIdx != -1){

          allocation [i] = best Idx;
          blocks [best Idx] -= porocessess [i] ;
        printf ("Process %.d allocated to block %.d (size %.d)\n", i+1,
                        bestIdx+1, blocks [bestIdx]);}

      else{
            printf("Process %d could not be allocated \n", i+1);

      }
   }
}

int main(){

      int m,n;
      printf("Enter the number of memory blocks :");
      scanf("%.d", &m);
      printf("Enter the number of processes:");
      scanf ("%.d",&n);
      int blocks[m], processess[n];
      printf( "Enter the sizes of %d memory blocks : \n", m);
      for (int i=0; i<m; i++){
              scanf("%d", & blocks[i]);
      }
      printf("Enter the sizes of %d processes: \n", n);

      for(int i=0; i<n; i++){
              scanf(".%d", &processes[i]);
      }
      bestFit (blocks, m, processes, n);
      return 0;
}
```

Input:

```
Enter the number of memory blocks: 5

Enter the number of processes: 4

Enter the sizes of 5 memory blocks:

10  20  30  40  50

Enter the sizes of 4 processes

5  10  20  25
```

output

```
Process No.        Process size        Block No.

    1                  5                   1

    2                  10                  2

    3                  10                  3

    4                  15                  4
```

Sample Output:

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

[illegible]

Result:

Thus, Best Fit memory allocation techique executed successfully.

Ex. No.: 10b)
Date: 10-04-25

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:

```
# include <stdio.h>

# define Max 25

void first Fit( int b[] , int f[] , int  nb, int nf){
             (frag of each file)            (store block no)
      int frag[MAX], bf[MAX], ff[MAX];
                            (block is T/F)
      int  i,j;

      int  highest=0;

       for (i=0; i< nb, i++){

             bf[i]=0;

       }

       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;
            highest =1;
            break;
        }
    }
    if (highest ==0){
            ff[i]=-1;
            frag[i]=-1;
        }
        highest =0;
    }
    printf("\n File no. \t File size \t Block No. \t Block size \t Fragmentation

    for (i=0; i<nf; i++){
            printf("%d \t \t %d \t\t", i+1, f[i]);
            if  (ff[i] != -1){
                    printf("%d \t \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("Not Allocated \t - \t \t - \n");
                }
            }
    }
```

```
                        → size of blocks    → size of files
int main(){
    int b[max], f[MAX];
    int nb, nf;
    int i;
    printf("Enter the number of memory blocks: ");
    scanf("%d", &nb);

    printf("Enter the number of files: ");
    scanf("%d", &nf);
    printf("\n Enter the sizes of blocks : \n");
    for (i=0; i<nb; i++){ printf("Block %d : ", i+1);
                          scanf("%d", &b[i]);

    }

    printf("\n Enter the sizes of the files: \n");
    for(i=0; i<nf; i++){
            printf("File %d : ", i+1);
            scanf("% .d", &f[i]);
    }
    firstFit (b, f, nb, nf);
    return 0;
}
```

Input:

Enter the number of memory blocks: 5

Enter the number of files: 4

Enter the sizes of the blocks:

Block 1: 100

Block 2: 200

Block 3: 300

Block 4: 150

Block 5: 500

Enter the sizes of the files:

Files 1: 120

File 2: 300

File 3: 150

File 4: 200

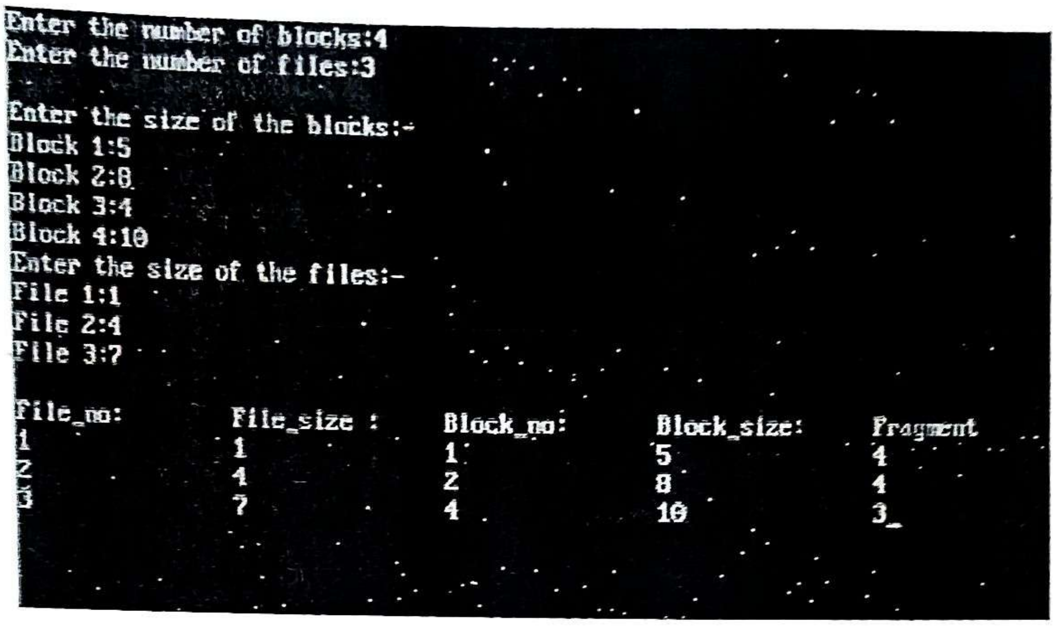
File 1 $\rightarrow$ block 1 ×
block 2 ✓
(200 - 120 = 8) Fragment

File2 $\rightarrow$ block 1 ×
block 2 ×
block 3 ✓ (300 - 300 = 0

Output

File No.	File size	Block No	Block size	Fragmentation
1	120	2	100	80 ~~Not allocated~~
2	300	3	300	0
3	150	4	150	0
4	200	5	200	300

Sample 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
3               7               4               10              3_
```

[illegible]

Result:

Thus, memory allocation methods for fixed position using first fit executed successfully.