Ex. No.: 10a)
Date: 11/4/25

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>
# include <stdbool.h>
int  main(){
        int  n,m;
        printf("Enter the no. of processes and blocks:");
        scanf(" %d  %d", &n ,&m);
        int  p[n],b[m];
        printf(" Enter the sizes of processes :\n");
        for (int  i=0; i<n ;i++)
                scanf(" %d", & p[i]);
        printf(" Enter the sizes of memory blocks\n:");
        for (int i=0 ;i<m ;i++){
                scanf("%d" , & b[i]);
        }
        int f[n], f2[m];
        for (int  i=0 ;i<n; i++)
                f[i]=-1;
for( int i=0; i<m; i++){
        f2[i]=0;
}
for (int i=0; i<n; i++){
            int k=-1;
            for (int j=0 ; j<m ; j++){
                if (b[i]>=p[i]  && f2[i]==0){
                    if (k==-1  || b[k]>b[j]){
                            k=j;
                    }
                }
            }
            if (k!=-1){
                f[i]=k;
                f[k]=1;
            }
    }
    printf ("process No    Process Size   Block No\n");
    for (int i=0; i<n; i++){
            if (f[i]!=-1)
                printf("%d  %d      %d\n", i+1, p[i], f[i]+1);

            else
                printf("%d  %d   %s\n", i+1, P[i], "Not Allocated");
    }
}
```

Sample Output:

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

Result:

Thus ~~[illegible]~~ the C program for Best fit was successfully executed.

[illegible]

Ex. No.: 10b)
Date: 11/4/25

FIRST FIT

Aim:

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

Algorithm:

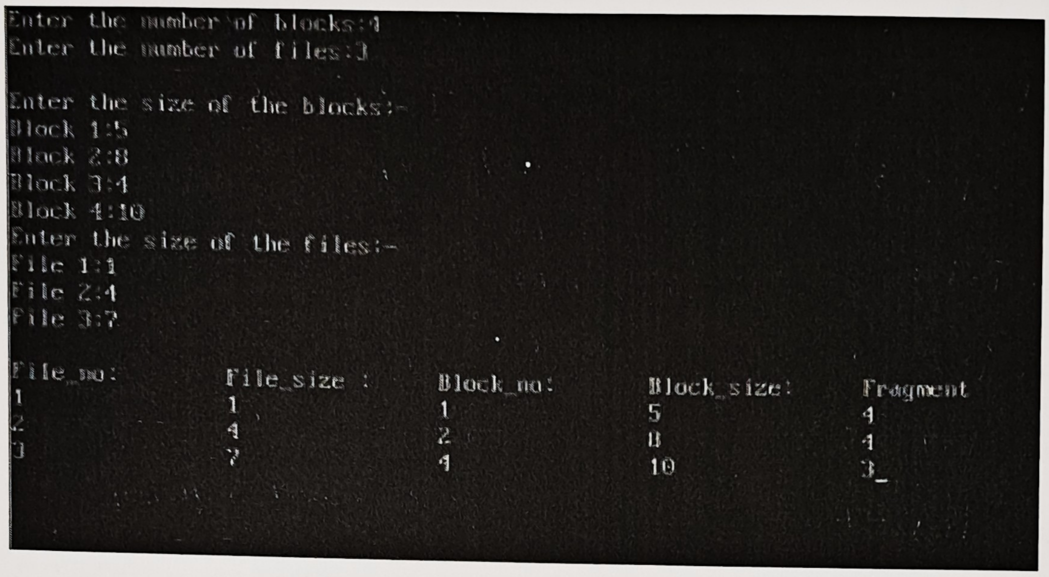
1. Define the max as 25.
2: Declare the variable frag[max],b[max],f[max],i,j,nb,nf,temp, highest=0, bf[max],ff[max]. 3: Get the number of blocks,files,size of the blocks using for loop.
4: In for loop check bf[j]!=1, if so temp=b[j]-f[i]
5: Check highest

Program Code:

```
#include <stdio.h>
# include <stdbool.h>

int main (){
    int n,m;
    printf("Enter the no. of processes and blocks:");
    scanf(" %d %d", &n, &m);

    int p[n], b[m];
    printf ("Enter the ~~sizes~~ of the processes:\n");
    for (int i=0; i<n; i++)
        scanf("%d", &p[i]);

    printf ("Enter the sizes of memory blocks:");
    for (int i=0; i<m ;i++){
        scanf("%d" , &b[i]);
    }
    int f[n], fs[m];
```

```
for (int i=0; i<n: i++){
        f[i]=-1;
}
for (int i=0; i<m; i++){
        f2[i]=0;
}
for (int i=0 ;i<n;i+ ){
        for (int j=0; j<m; j++){
                if (b[i]>=p[i]   && f2[i]==0){
                        f[i]=j;
                        f2[ j ]=1;
                        break;
                }
        }
}
printf ("Process No.  Process Size   Block No   Block Size
                                              fragment");
for (int i=0; i<n; i++){
        if (f[i]!=-1){
                                    %d %d
                printf ("%d   %d  ^%d\n", i+1, p[i], f[i]+1,
                                 b[f[i]], b[f[i]]- p[i]);
        }
        else{
                printf ("%d  %d  %s\n", i+1, p[i], "Not
                                               Allocated");
        }
}
```

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 the c program for first fit is executed sucessfully.

Output:

```
Enter the number of processes and blocks : 3   3

Enter the sizes of the processes:

100      200      300

Process No     Process Size     Block No
1              100              2
2              200              1
3              300              Not Allocated
```

Output:

```
Enter the number of processes And blocks: 4   5

Enter the sizes of the processes:

212     414     112     426

Enter the sizes of the memory blocks:

100     500     200     300     600

process No.     Process Size     Block No     Block Size     Fragment
                                                500            288
1                212              2

2                417              5             600            183

3                112              3             200            88

4                426              Not Allocated
```