Experiment No. 7

Name : Rinoy Kuriyakose
Roll No: 56

Aim:

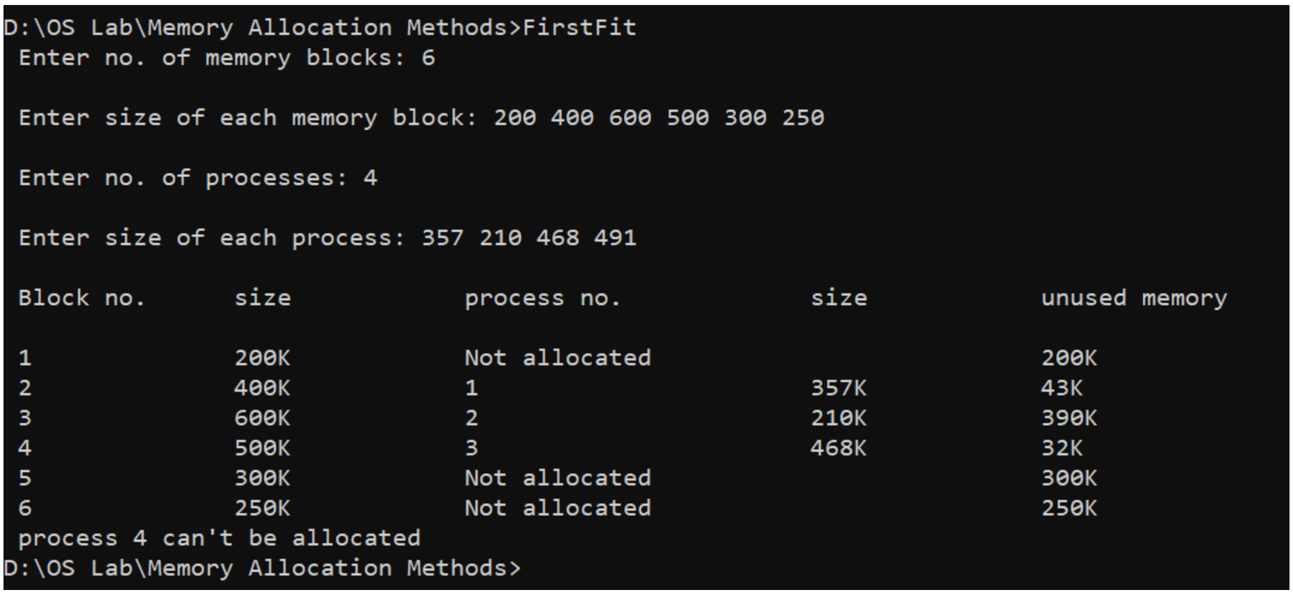
Implementation of the Memory Allocation Methods for fixed partition*

a) First Fit b) Worst Fit c) Best Fit

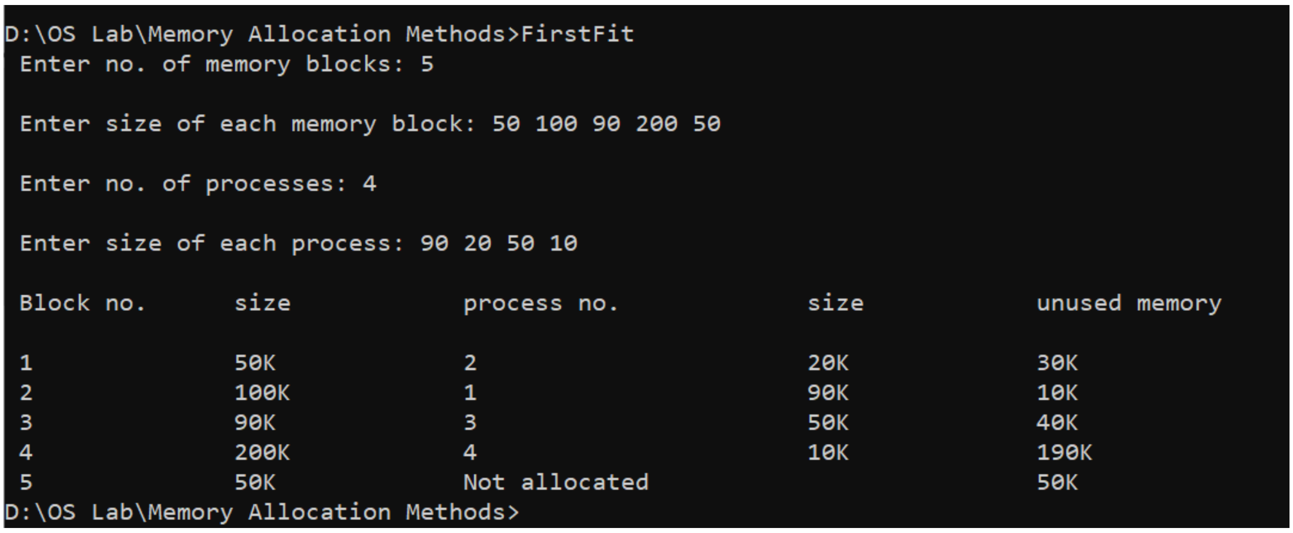
a) First Fit Program:

```
#include<stdio.h>
void main(){
    int block[50], process[50], bsize, psize, iscompleted[50], allocation[50], i, j;
    printf(" Enter no. of memory blocks: ");
    scanf("%d", &bsize);
    printf("\n Enter size of each memory block: ");
    for(i = 0; i < bsize; i++){
        scanf("%d", &block[i]);
    }
    printf("\n Enter no. of processes: ");
    scanf("%d", &psize);
    printf("\n Enter size of each process: ");
    for(i = 0; i < psize; i++){
        scanf("%d", &process[i]);
    }
    for(i = 0; i < bsize; i++){
        allocation[i] = -1;
    }
    for(i = 0; i < psize; i++){
        iscompleted[i] = 0;
    }
    for(i = 0; i < psize; i++){
        for(j = 0; j < bsize; j++){
            if(allocation[j] == -1 && block[j] >= process[i]){
                allocation[j] = i;
                iscompleted[i] = 1;
                break;
            }
        }
    }
    printf("\n Block no.\tsize\t\tprocess no.\t\tsize\t\tunused memory\n");
    for(i = 0; i < bsize; i++){
        printf("\n %d\t\t%dK\t\t", i+1, block[i]);
        if(allocation[i] != -1){
            printf("%d\t\t\t%dK\t\t%dK",allocation[i]+1,process[allocation[i]],block[i]-
process[allocation[i]]);
        }else{
            printf("Not allocated\t\t\t\t%dK",block[i]);
        }
    }
    for(i = 0; i < psize; i++){
        if(iscompleted[i] != 1){
            printf("\n process %d can't be allocated",i+1);
        }
    }

}
```

Output:



```
D:\OS Lab\Memory Allocation Methods>FirstFit
 Enter no. of memory blocks: 6

 Enter size of each memory block: 200 400 600 500 300 250

 Enter no. of processes: 4

 Enter size of each process: 357 210 468 491

 Block no.      size            process no.             size            unused memory

 1              200K            Not allocated                           200K
 2              400K            1                       357K            43K
 3              600K            2                       210K            390K
 4              500K            3                       468K            32K
 5              300K            Not allocated                           300K
 6              250K            Not allocated                           250K
 process 4 can't be allocated
D:\OS Lab\Memory Allocation Methods>
```

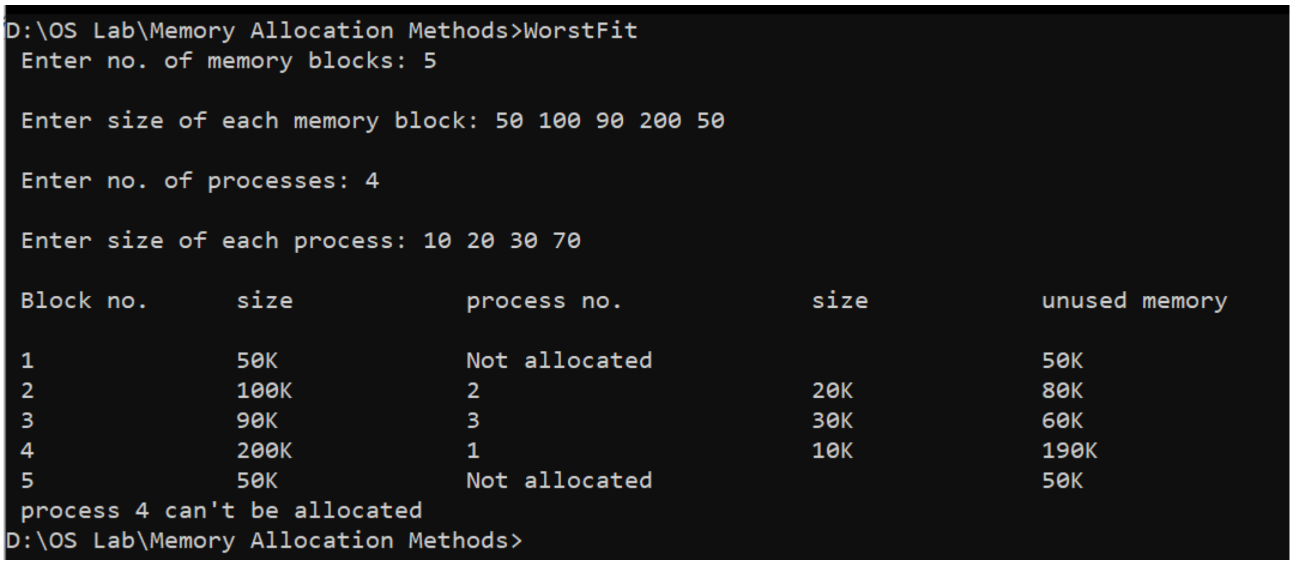


```
D:\OS Lab\Memory Allocation Methods>FirstFit
 Enter no. of memory blocks: 5

 Enter size of each memory block: 50 100 90 200 50

 Enter no. of processes: 4

 Enter size of each process: 90 20 50 10

 Block no.      size            process no.             size            unused memory

 1              50K             2                       20K             30K
 2              100K            1                       90K             10K
 3              90K             3                       50K             40K
 4              200K            4                       10K             190K
 5              50K             Not allocated                           50K
D:\OS Lab\Memory Allocation Methods>
```

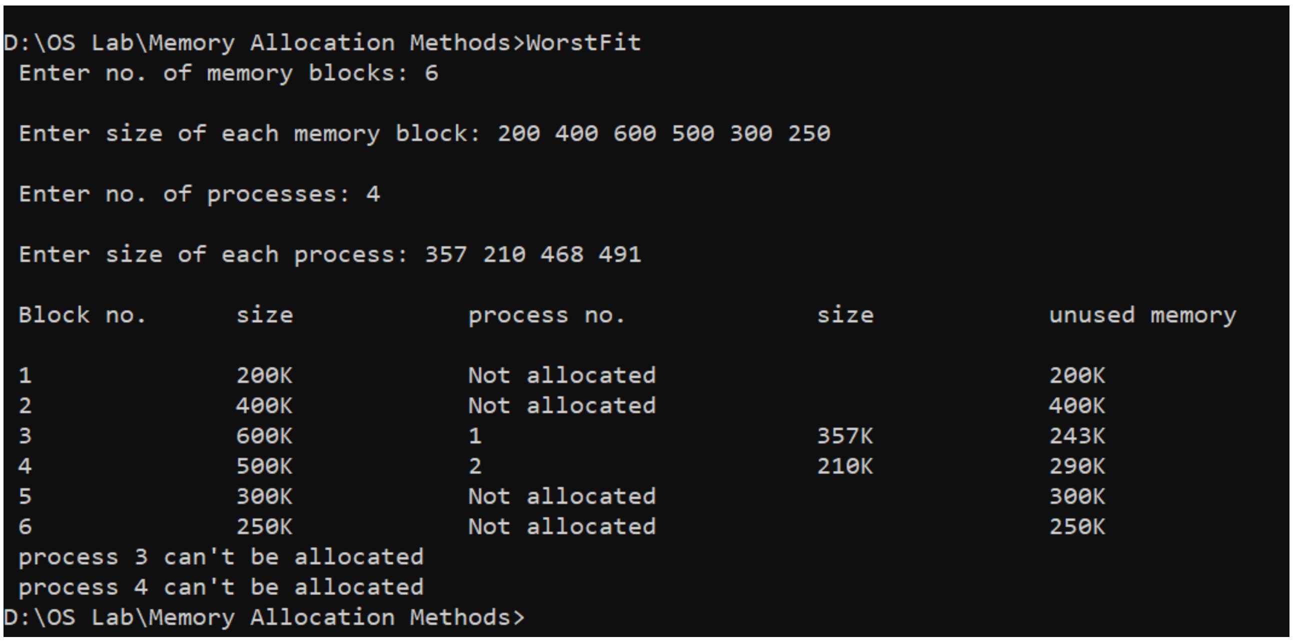
b) Worst Fit Program:

```
#include<stdio.h>
void main(){
    int block[50], process[50], bsize, psize, iscompleted[50], allocation[50],i, j,index;
    printf(" Enter no. of memory blocks: ");
    scanf("%d", &bsize);
    printf("\n Enter size of each memory block: ");
    for(i = 0; i < bsize; i++){
        scanf("%d", &block[i]);
    }
    printf("\n Enter no. of processes: ");
    scanf("%d", &psize);
    printf("\n Enter size of each process: ");
    for(i = 0; i < psize; i++){
        scanf("%d", &process[i]);
    }
    for(i = 0; i < bsize; i++){
        allocation[i] = -1;
    }
    for(i = 0; i < psize; i++){
        iscompleted[i] = 0;
    }
    for(i=0;i<psize;i++){
        index =-1;
        for(j=0;j<bsize;j++){
            if((block[j]>=process[i])&&(allocation[j]==-1)){
                if(index==-1){
                    index=j;
                }else if( block[index] < block[j]){
                    index = j;
                }
            }
        }
        if(index!=-1){
            allocation[index] = i;
            iscompleted[i]=1;
        }
    }
    printf("\n Block no.\tsize\t\tprocess no.\t\tsize\t\tunused memory\n");
    for(i = 0; i < bsize; i++){
        printf("\n %d\t\t%dK\t\t", i+1, block[i]);
        if(allocation[i] != -1){
            printf("%d\t\t\t%dK\t\t%dK",allocation[i]+1,process[allocation[i]],block[i]-
process[allocation[i]]);
        }else{
            printf("Not allocated\t\t\t\t%dK",block[i]);
        }
    }
    for(i = 0; i < psize; i++){
        if(iscompleted[i] != 1){
            printf("\n process %d can't be allocated",i+1);
        }
    }

}
```

Output:



```
D:\OS Lab\Memory Allocation Methods>WorstFit
 Enter no. of memory blocks: 5

 Enter size of each memory block: 50 100 90 200 50

 Enter no. of processes: 4

 Enter size of each process: 10 20 30 70

 Block no.      size           process no.           size           unused memory

 1              50K            Not allocated                        50K
 2              100K           2                     20K            80K
 3              90K            3                     30K            60K
 4              200K           1                     10K            190K
 5              50K            Not allocated                        50K
 process 4 can't be allocated
D:\OS Lab\Memory Allocation Methods>
```

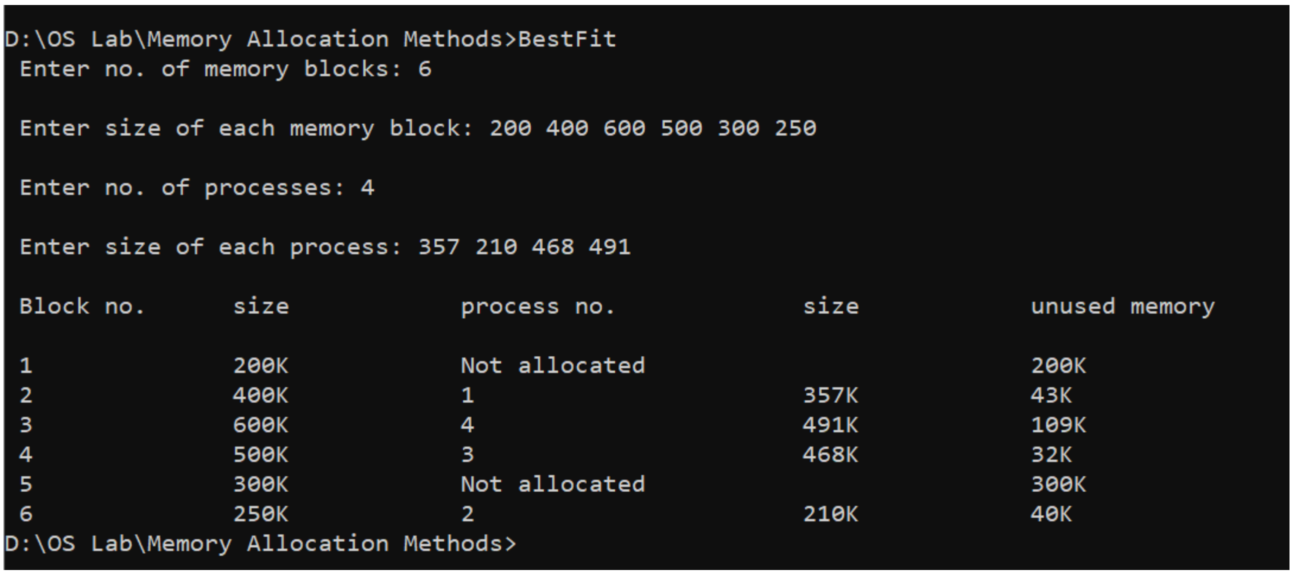


```
D:\OS Lab\Memory Allocation Methods>WorstFit
 Enter no. of memory blocks: 6

 Enter size of each memory block: 200 400 600 500 300 250

 Enter no. of processes: 4

 Enter size of each process: 357 210 468 491

 Block no.      size           process no.           size           unused memory

 1              200K           Not allocated                        200K
 2              400K           Not allocated                        400K
 3              600K           1                     357K           243K
 4              500K           2                     210K           290K
 5              300K           Not allocated                        300K
 6              250K           Not allocated                        250K
 process 3 can't be allocated
 process 4 can't be allocated
D:\OS Lab\Memory Allocation Methods>
```

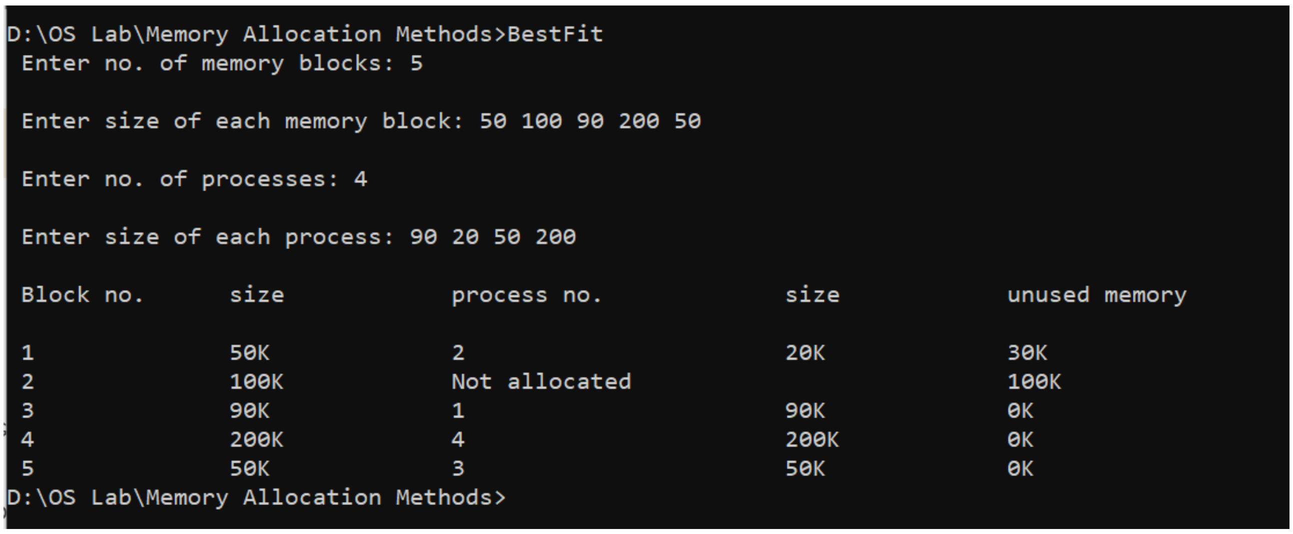
c) Best Fit Program:

```
#include<stdio.h>
void main(){
    int block[50], process[50], bsize, psize, iscompleted[50], allocation[50],temp,i, j,index=-1;
    printf(" Enter no. of memory blocks: ");
    scanf("%d", &bsize);
    printf("\n Enter size of each memory block: ");
    for(i = 0; i < bsize; i++){
        scanf("%d", &block[i]);
    }
    printf("\n Enter no. of processes: ");
    scanf("%d", &psize);
    printf("\n Enter size of each process: ");
    for(i = 0; i < psize; i++){
        scanf("%d", &process[i]);
    }
    for(i = 0; i < bsize; i++){
        allocation[i] = -1;
    }
    for(i = 0; i < psize; i++){
        iscompleted[i] = 0;
    }
    for(i=0;i<psize;i++){
        index =-1;
        for(j=0;j<bsize;j++){
            if((block[j]>=process[i])&&(allocation[j]==-1)){
                if(index==-1){
                    index=j;
                }else if( block[index] > block[j]){
                    index = j;
                }
            }
        }
        if(index!=-1){
            allocation[index] = i;
            iscompleted[i]=1;
        }
    }
    printf("\n Block no.\tsize\t\tprocess no.\t\tsize\t\tunused memory\n");
    for(i = 0; i < bsize; i++){
        printf("\n %d\t\t%dK\t\t", i+1, block[i]);
        if(allocation[i] != -1){
            printf("%d\t\t\t%dK\t\t%dK",allocation[i]+1,process[allocation[i]],block[i]-
process[allocation[i]]);
        }else{
            printf("Not allocated\t\t\t\t%dK",block[i]);
        }
    }
    for(i = 0; i < psize; i++){
        if(iscompleted[i] != 1){
            printf("\n process %d can't be allocated",i+1);
        }
    }

}
```

Output:



```
D:\OS Lab\Memory Allocation Methods>BestFit
 Enter no. of memory blocks: 6

 Enter size of each memory block: 200 400 600 500 300 250

 Enter no. of processes: 4

 Enter size of each process: 357 210 468 491

 Block no.      size            process no.             size            unused memory

 1              200K            Not allocated                           200K
 2              400K            1                       357K            43K
 3              600K            4                       491K            109K
 4              500K            3                       468K            32K
 5              300K            Not allocated                           300K
 6              250K            2                       210K            40K
D:\OS Lab\Memory Allocation Methods>
```



```
D:\OS Lab\Memory Allocation Methods>BestFit
 Enter no. of memory blocks: 5

 Enter size of each memory block: 50 100 90 200 50

 Enter no. of processes: 4

 Enter size of each process: 90 20 50 200

 Block no.      size            process no.             size            unused memory

 1              50K             2                       20K             30K
 2              100K            Not allocated                           100K
 3              90K             1                       90K             0K
 4              200K            4                       200K            0K
 5              50K             3                       50K             0K
D:\OS Lab\Memory Allocation Methods>
```