Don Bosco Institute of Technology,Kurla
Academic Year 2019-20

EXPERIMENT NO: 7

Title: A program to simulate memory allocation policies

By -

Shawn Louis

Roll No : 31

Batch : B

Class:S.E Comps(Sem IV) **Lecturer:Sejal.Chopra**

Subject: PA Lab

EXPERIMENT NO: 7

Simulate memory allocation policies

AIM	Write a program to simulate memory allocation policies: a)First-fit algorithm ;b)Best-fit algorithm ;c)Next-fit algorithm ;d)Worst-fit algorithm
LEARNING OBJECTIVE	To implement various memory allocation policies.
LEARNING OUTCOME	Student will be able to understand how new processes are allocated memory.
LAB OUTCOME	CSL 403.1: Ability to compile a code for computer operations.
PROGRAM OUTCOME	PO1-1, PO5-2, PO8-3, PO9-3, PO12-2, PSO1-2
BLOOM'S TAXONOMY LEVEL	Remember, Understand
THEORY	**1. First Fit Algorithm** First Fit algorithm scans the linked list and whenever it finds the first big enough hole to store a process, it stops scanning and load the process into that hole. This procedure produces two partitions. Out of them, one partition will be a hole while the other partition will store the process. First Fit algorithm maintains the linked list according to the increasing order of starting index. This is the simplest to implement among all the algorithms and produces bigger holes as compare to the other algorithms. **2. Next Fit Algorithm** Next Fit algorithm is similar to First Fit algorithm except the fact that, Next fit scans the linked list from the node where it previously allocated a hole. Next fit doesn't scan the whole list, it starts scanning the list from the next node. The idea behind the next fit is the fact that the list has been scanned once therefore the probability of finding the hole is larger in the remaining part of the list. Experiments over the algorithm have shown that the next fit is not better then the first fit. So it is not being used these days in most of the cases.

3. Best Fit Algorithm

The Best Fit algorithm tries to find out the smallest hole possible in the list that can accommodate the size requirement of the process.

Using Best Fit has some disadvantages.

1. 1. It is slower because it scans the entire list every time and tries to find out the smallest hole which can satisfy the requirement the process.
2. Due to the fact that the difference between the whole size and the process size is very small, the holes produced will be as small as it cannot be used to load any process and therefore it remains useless. Despite of the fact that the name of the algorithm is best fit, It is not the best algorithm among all.

4. Worst Fit Algorithm

The worst fit algorithm scans the entire list every time and tries to find out the biggest hole in the list which can fulfill the requirement of the process.

Despite of the fact that this algorithm produces the larger holes to load the other processes, this is not the better approach due to the fact that it is slower because it searches the entire list every time again and again.

SOFTWARE USED

C/C++/Java

STEPS TO EXECUTE THE PROGRAM

1. Develop a memory model by allocating some processes of a specific size in memory.
2. Consider an example starting from lower memory locations:

Memory Contents	Size in memory
Process P1	3K
Free space 1	4K
Process P2	6K
Free space 2	5K
Process P3	1K

Free space 3	2K

3. Ask the user to enter the new process which will arrive and with its size (example:P4 :2K)
4. The user should also provide with which algorithm it is going to implement the memory allocation.
5. As per the algorithm used it should accommodate respective free space.

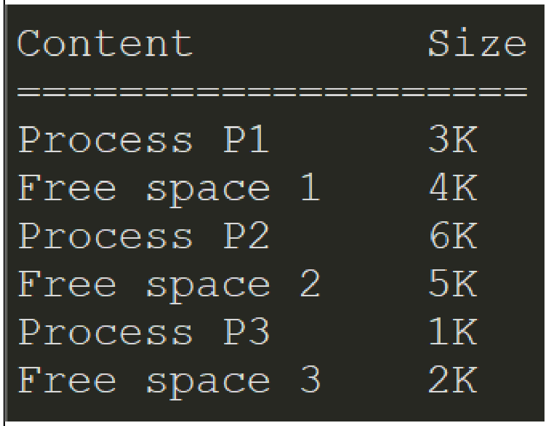
CODE

Source Code:

```
name=["Process P1","Free space 1","Process P2","Free space 2","Process P3","Free space 3"]
mem=[3,4,6,5,1,2]

def printTable():
    global name,mem
    print("Content\t\tSize")
    print("====================")
    for (x,y) in zip(name,mem):
        print("{}\t{}K\t".format(x,y))

def inputs():
    global m
    print()
    m=int(input("Enter the size of Process P4 : "))
    print()
    print()
    print("1. First Fit")
    print("2. Best Fit")
    print("3. Next Fit")
    print("4. Worst Fit")
    print()
    global ch
    ch=int(input("Enter your choice : "))
    print()

def choice():
    global ch
    if(ch==1):
        firstFit()
    if(ch==2):
        bestFit()
    if(ch==3):
        nextFit()
    if(ch==4):
        worstFit()
```

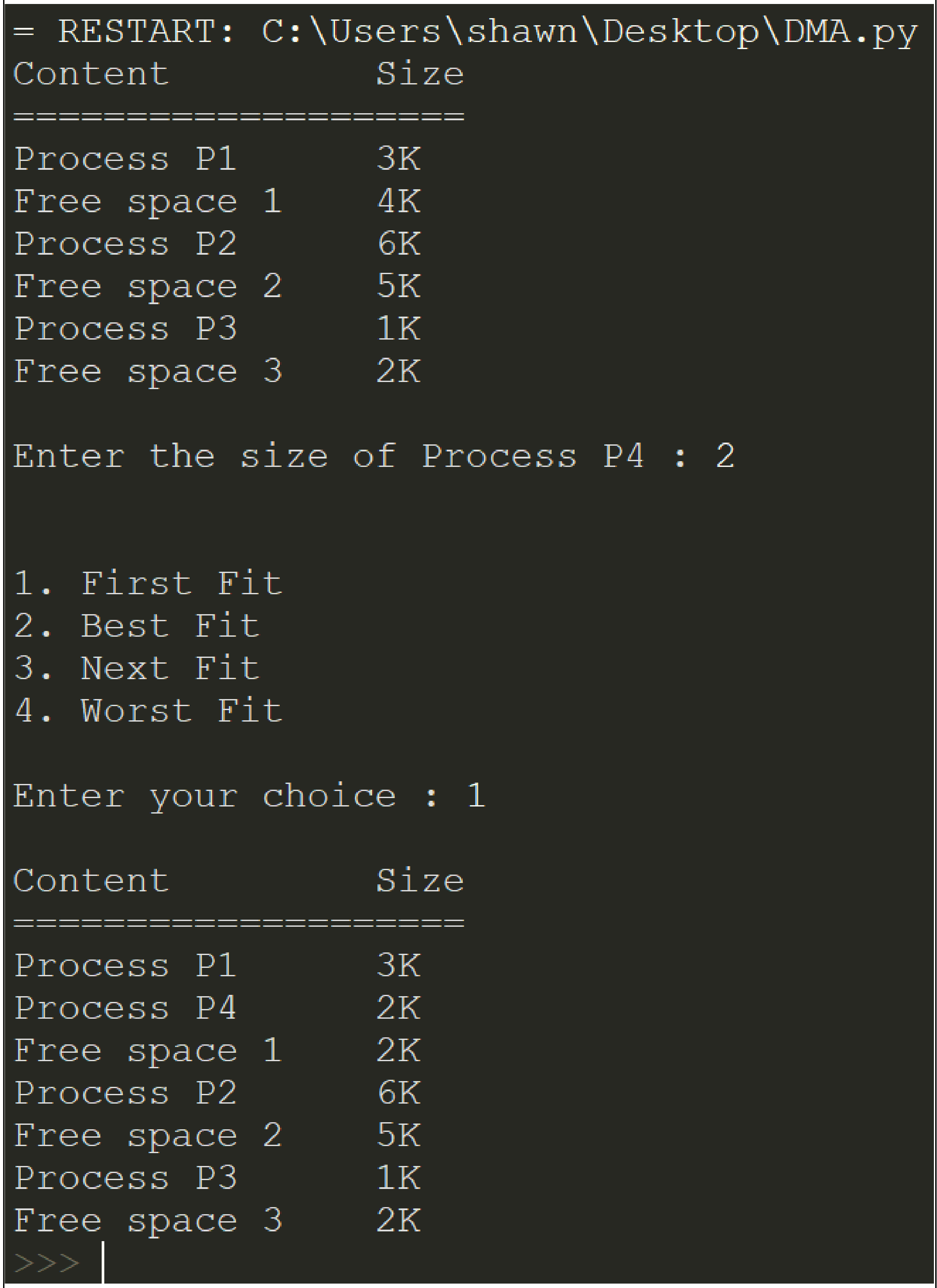
```
def firstFit():
    t=0
    global name,mem,m
    for (x,y) in zip(name,mem):
        if(x[0]=='F'):
            if y>=m:
                break
        t=t+1
    name.insert(t,"Process P4")
    mem.insert(t,m)
    mem[t+1]=mem[t+1]-m
    if mem[t+1]==0:
        del mem[t+1]
        del name[t+1]

def swap(list, pos1, pos2):

    list[pos1], list[pos2] = list[pos2], list[pos1]
    return list

def nextFit():
    t=0
    global name,mem,m
    name.reverse()
    mem.reverse()
    for (x,y) in zip(name,mem):
        if(x[0]=='F'):
            if y>=m:
                break
        t=t+1
    name.insert(t,"Process P4")
    mem.insert(t,m)
    mem[t+1]=mem[t+1]-m
    if mem[t+1]==0:
        del mem[t+1]
        del name[t+1]
    else:
        swap(name,t,t+1)
        swap(mem,t,t+1)
    name.reverse()
    mem.reverse()

def bestFit():
    global name,mem,m
    t=1000
    for (x,y) in zip(name,mem):
        if(x[0]=='F'):
            if y>=m and y<t:
                t=y
    t=mem.index(t)
    name.insert(t,"Process P4")
    mem.insert(t,m)
    mem[t+1]=mem[t+1]-m
    if mem[t+1]==0:
        del mem[t+1]
        del name[t+1]

def worstFit():
    global name,mem,m
    t=-1000
    for (x,y) in zip(name,mem):
        if(x[0]=='F'):
            if y>=m and y>t:
                t=y
    t=mem.index(t)
    name.insert(t,"Process P4")
    mem.insert(t,m)
    mem[t+1]=mem[t+1]-m
    if mem[t+1]==0:
        del mem[t+1]
        del name[t+1]

printTable()
inputs()
choice()
printTable()
```

Pre allocation table (common to all):



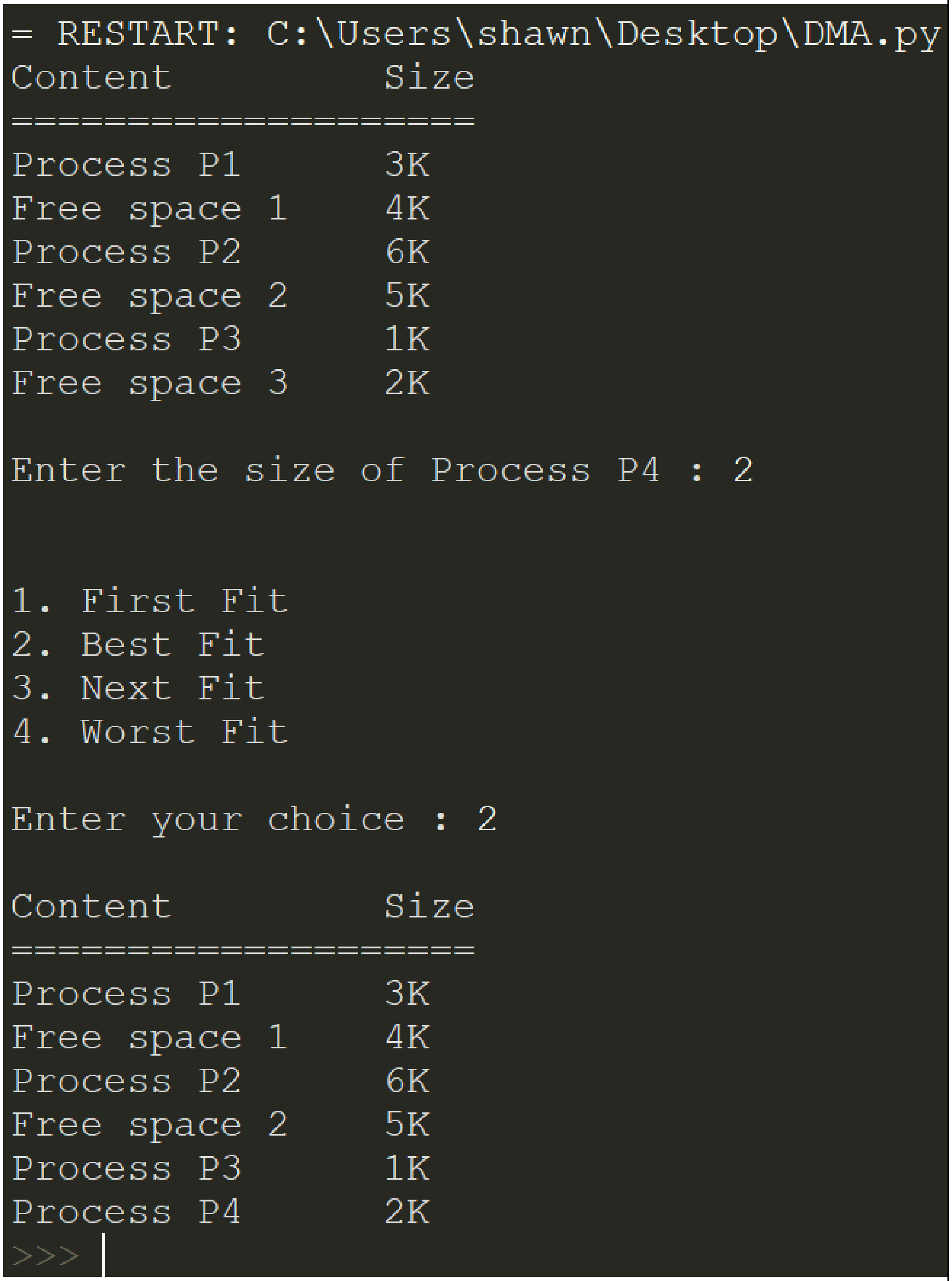
```
Content          Size
=====================
Process P1       3K
Free space 1     4K
Process P2       6K
Free space 2     5K
Process P3       1K
Free space 3     2K
```

First fit:



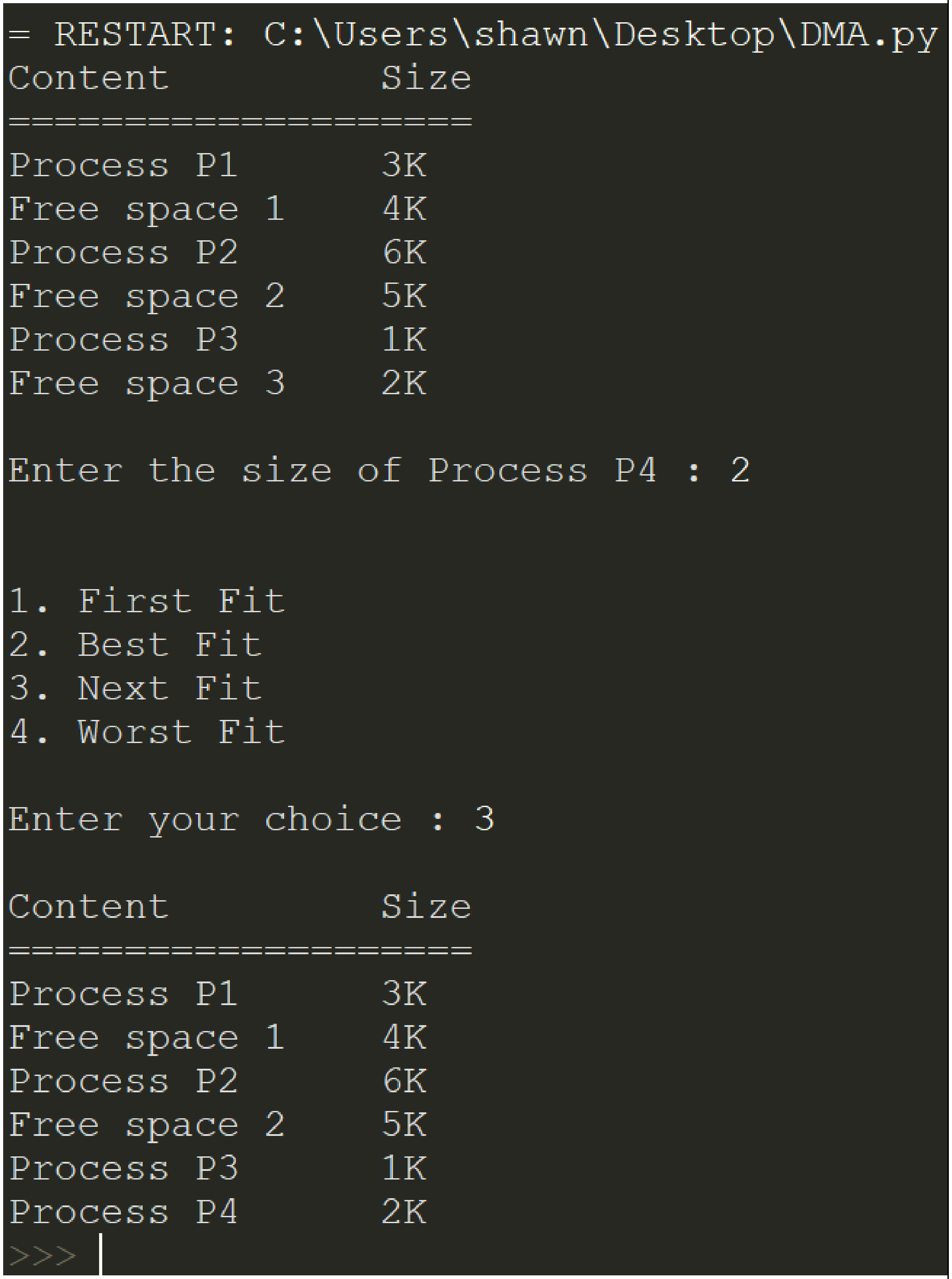
```
= RESTART: C:\Users\shawn\Desktop\DMA.py
Content             Size
====================
Process P1          3K
Free space 1        4K
Process P2          6K
Free space 2        5K
Process P3          1K
Free space 3        2K

Enter the size of Process P4 : 2



1. First Fit
2. Best Fit
3. Next Fit
4. Worst Fit

Enter your choice : 1

Content             Size
====================
Process P1          3K
Process P4          2K
Free space 1        2K
Process P2          6K
Free space 2        5K
Process P3          1K
Free space 3        2K
>>> 
```

Best fit:



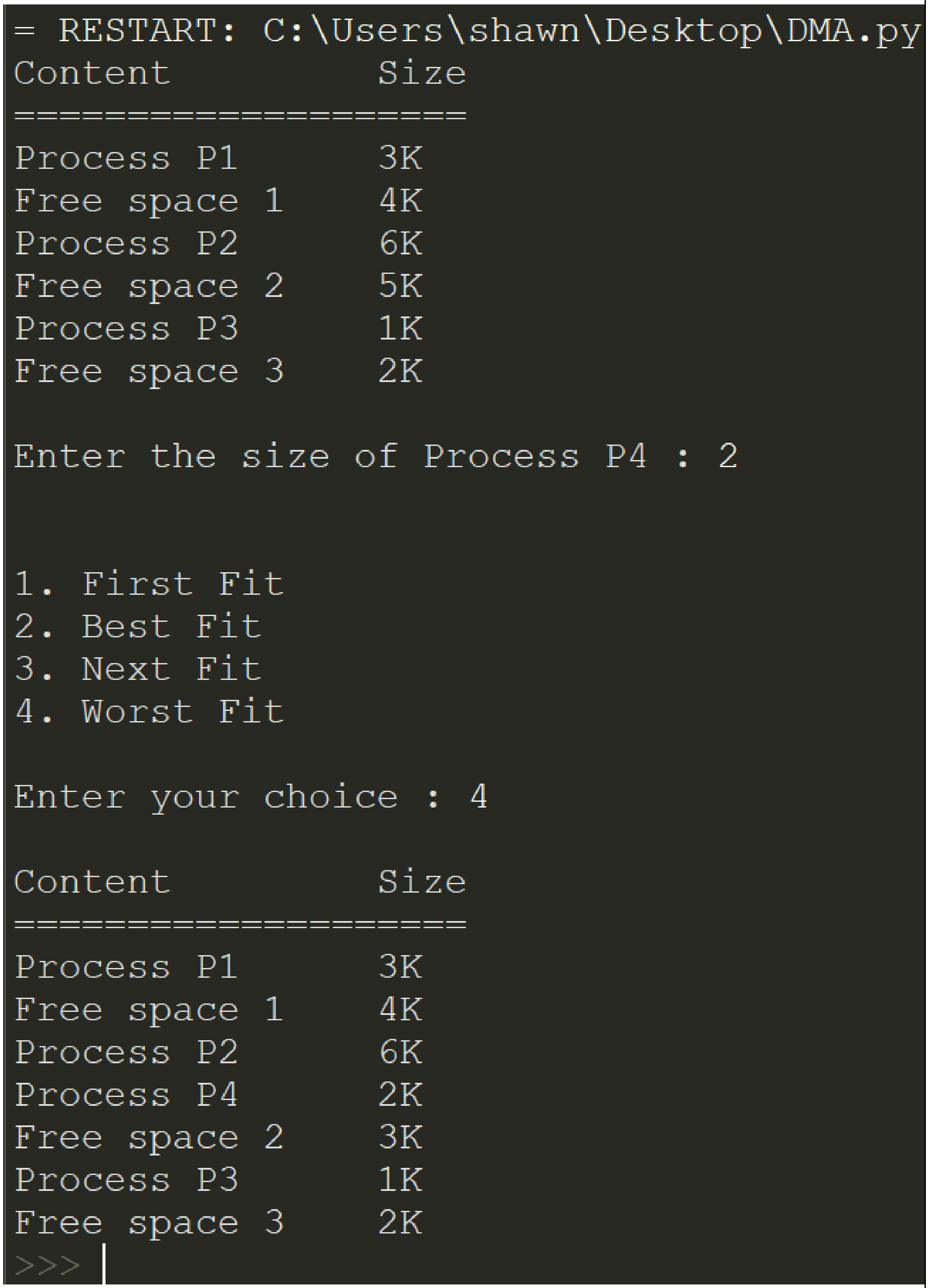
```
= RESTART: C:\Users\shawn\Desktop\DMA.py
Content         Size
====================
Process P1      3K
Free space 1    4K
Process P2      6K
Free space 2    5K
Process P3      1K
Free space 3    2K

Enter the size of Process P4 : 2


1. First Fit
2. Best Fit
3. Next Fit
4. Worst Fit

Enter your choice : 2

Content         Size
====================
Process P1      3K
Free space 1    4K
Process P2      6K
Free space 2    5K
Process P3      1K
Process P4      2K
>>> 
```

Next fit:



```
= RESTART: C:\Users\shawn\Desktop\DMA.py
Content         Size
====================
Process P1      3K
Free space 1    4K
Process P2      6K
Free space 2    5K
Process P3      1K
Free space 3    2K

Enter the size of Process P4 : 2



1. First Fit
2. Best Fit
3. Next Fit
4. Worst Fit

Enter your choice : 3

Content         Size
====================
Process P1      3K
Free space 1    4K
Process P2      6K
Free space 2    5K
Process P3      1K
Process P4      2K
>>> 
```

Worst fit:



```
= RESTART: C:\Users\shawn\Desktop\DMA.py
Content            Size
====================
Process P1         3K
Free space 1       4K
Process P2         6K
Free space 2       5K
Process P3         1K
Free space 3       2K

Enter the size of Process P4 : 2


1. First Fit
2. Best Fit
3. Next Fit
4. Worst Fit

Enter your choice : 4

Content            Size
====================
Process P1         3K
Free space 1       4K
Process P2         6K
Process P4         2K
Free space 2       3K
Process P3         1K
Free space 3       2K
>>> 
```

CONCLUSION

Hence we learnt how to implement the different fit algorithms using python code and saw the different outputs to understand the algorithms

REFERENCES	1. William Stallings, "Computer Organization and Architecture: Designing for Performance",Pearson Publication, 10 th Edition, 2013 2. B. Govindarajulu, "Computer Architecture and Organization: Design Principles and Applications", Second Edition, McGraw-Hill (India)