EXPERIMENT NO: 7

Title: A program to simulate memory allocation policies

Name:Hayden Cordiero SECOMPS RollNo:05

EXPERIMENT NO: 7

Simulate memory allocation policies

<table>
<tr><td>AIM</td><td>Write a program to simulate memory allocation policies: a)First-fit algorithm;b)Bestfit algorithm ;c)Nextfit algorithm ;d)Worstfit algorithm</td></tr>
<tr><td>LEARNING OBJECTIVE</td><td>To implement various memory allocation policies.</td></tr>
<tr><td>LEARNINGOUTCOME</td><td>Studentwillbeabletounderstandhownewprocessesareallocated memory.</td></tr>
<tr><td>LAB OUTCOME</td><td>CSL 403.1: Ability to compile a code for computer operations.</td></tr>
<tr><td>PROGRAM OUTCOME</td><td>PO11,
PO52,
PO83,
PO9-3,PO12-2, PSO1-2</td></tr>
<tr><td>BLOOM'S TAXONOMY LEVEL</td><td>Remember,
Understand</td></tr>
<tr><td>THEORY</td><td>Note for students:
Studentsaresupposetowritetheoryrelatedtomemoryallocation policiesandexplainitsworking,clearlydescribingitsprosandcons.</td></tr>
<tr><td>SOFTWARE USED</td><td>C/C++/Java</td></tr>
<tr><td>STEPS TO EXECUTE THE PROGRAM</td><td>1. Developamemorymodelbyallocatingsomeprocessesofaspecific size inmemory.
2. Consideranexamplestartingfromlowermemorylocations:
<table><tr><th>Memory Contents</th><th>Size in memory</th></tr><tr><td>Process P1</td><td>3K</td></tr><tr><td>Free space 1</td><td>4K</td></tr><tr><td>Process P2</td><td>6K</td></tr></table></td></tr>
</table>

	Free space 2	5K	
	Process P3	1K	

	Free space 3	2K	

3. Ask the user to enterthenewprocesswhichwillarriveandwithits size (example:P4:2K)
4. The usershouldalsoprovidewithwhichalgorithmitisgoingto implement the memoryallocation.
   Asperthealgorithmuseditshouldaccommodaterespectivefree space.

CODE

```
index=[1,2,3]
pro=[3,6,1]
fs=[4,5,2]
vc=[0,0,0]
p4=2

def FirstFit():
        for i in enumerate(fs):
            if(p4<=i[1]):
                pro.insert(i[0]+1,p4)
                fs[i[0]]=fs[i[0]]-p4
                fs.insert(i[0]+1,p4)
                index.insert(i[0]+1,4)
                vc.insert(i[0],1)
                vc[i[0]+1]=2
                break

        for i in range(len(pro)):
                if(vc[i]==1):
                        print('Process P'+str(index[i]),':',str(pro[i])+'k')
                        print('Process
P'+str(index[i+1]),':',str(pro[i+1])+'k')
                        if(fs[i]!=0):
                                print('New FreeS',':',str(fs[i])+'k')
                        i+=1
                elif(vc[i]!=2):
                        print('Process P'+str(index[i]),':',str(pro[i])+'k')
                        if(fs[i]!=0):
                                print('Free
Space'+str(index[i]),':',str(fs[i])+'k')




def Next_fit():
```

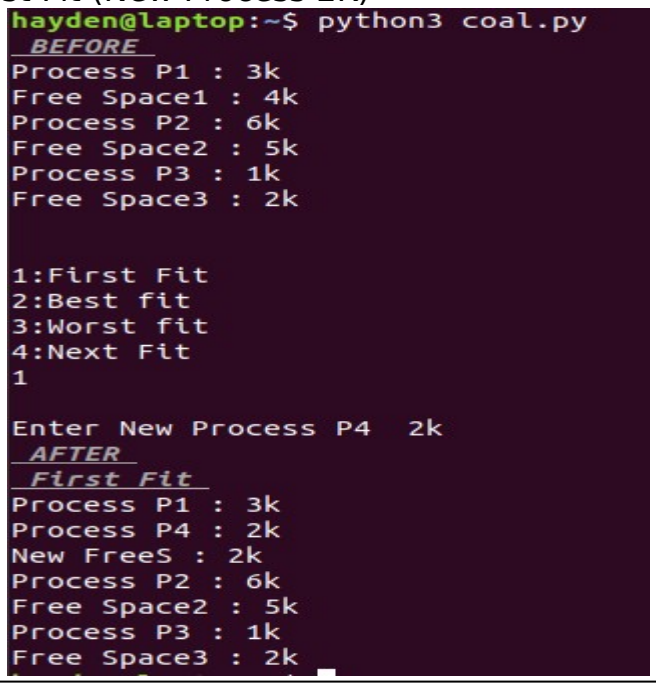
```
        if(fs[-1]-p4>0):
                i=(len(fs)-1,fs[-1])
                pro.insert(i[0]+1,p4)
                fs[i[0]]=fs[i[0]]-p4
                fs.insert(i[0]+1,p4)
                index.insert(i[0]+1,4)
                vc.insert(i[0],1)
                vc[i[0]+1]=2
                for i in range(len(pro)):
                        if(vc[i]==1):
                                print('Process
P'+str(index[i]),':',str(pro[i])+'k')
                                print('Process
P'+str(index[i+1]),':',str(pro[i+1])+'k')
                                if(fs[i]!=0):
                                        print('New FreeS',':',str(fs[i])+'k')
                                i+=1
                        elif(vc[i]!=2):
                                print('Process
P'+str(index[i]),':',str(pro[i])+'k')
                                if(fs[i]!=0):
                                        print('Free
Space'+str(index[i]),':',str(fs[i])+'k')

        else:
                FirstFit()









def Best_fit():
        min1=(0,fs[0])
        for i in enumerate(fs):
                if(i[1]<min1[1]):
                        min1=i
        i=min1
        pro.insert(i[0]+1,p4)
        fs[i[0]]=fs[i[0]]-p4
        fs.insert(i[0]+1,p4)
        index.insert(i[0]+1,4)
        vc.insert(i[0],1)
        vc[i[0]+1]=2
```

```
	for i in range(len(pro)):
		if(vc[i]==1):
			print('Process P'+str(index[i]),':',str(pro[i])+'k')
			print('Process
P'+str(index[i+1]),':',str(pro[i+1])+'k')
			if(fs[i]!=0):
				print('New FreeS',':',str(fs[i])+'k')
			i+=1
		elif(vc[i]!=2):
			print('Process P'+str(index[i]),':',str(pro[i])+'k')
			if(fs[i]!=0):
				print('Free
Space'+str(index[i]),':',str(fs[i])+'k')









def Worst_fit():
	tl=[]
	for i in (fs):
		tl.append(i-p4)

	i=(tl.index(max(tl)),max(tl))
	pro.insert(i[0]+1,p4)
	fs[i[0]]=fs[i[0]]-p4
	fs.insert(i[0]+1,p4)
	index.insert(i[0]+1,4)
	vc.insert(i[0],1)
	vc[i[0]+1]=2


	for i in range(len(pro)):
		if(vc[i]==1):
			print('Process P'+str(index[i]),':',str(pro[i])+'k')
			print('Process
P'+str(index[i+1]),':',str(pro[i+1])+'k')
			if(fs[i]!=0):
				print('New FreeS',':',str(fs[i])+'k')
			i+=1
		elif(vc[i]!=2):
			print('Process P'+str(index[i]),':',str(pro[i])+'k')
```

```
                    if(fs[i]!=0):
                        print('Free
Space'+str(index[i]),':',str(fs[i])+'k')




ask=input('1:First Fit\n2:Best fit\n3:Worst fit\n4:Next Fit')
p4=input('Enter New Process P4  ')
p4=int(p4[:len(p4)-1])
if(ask=='1'):
    print('\033[4m''\033[1m''\033[2m''\033[3m' +" First Fit "+'\
033[0;0m')
    FirstFit()
if(ask=='2'):
    print('\033[4m''\033[1m''\033[2m''\033[3m' +" Best Fit "+'\
033[0;0m')
    Best_fit()
if(ask=='3'):
    print('\033[4m''\033[1m''\033[2m''\033[3m' +" Worst Fit
"+'\033[0;0m')
    Worst_fit()
if(ask=='4'):
    print('\033[4m''\033[1m''\033[2m''\033[3m' +" Next Fit
"+'\033[0;0m')

    Next_fit()
```

OUTPUT:

First Fit (New Process 2K)



Best Fit (New Process 2K)

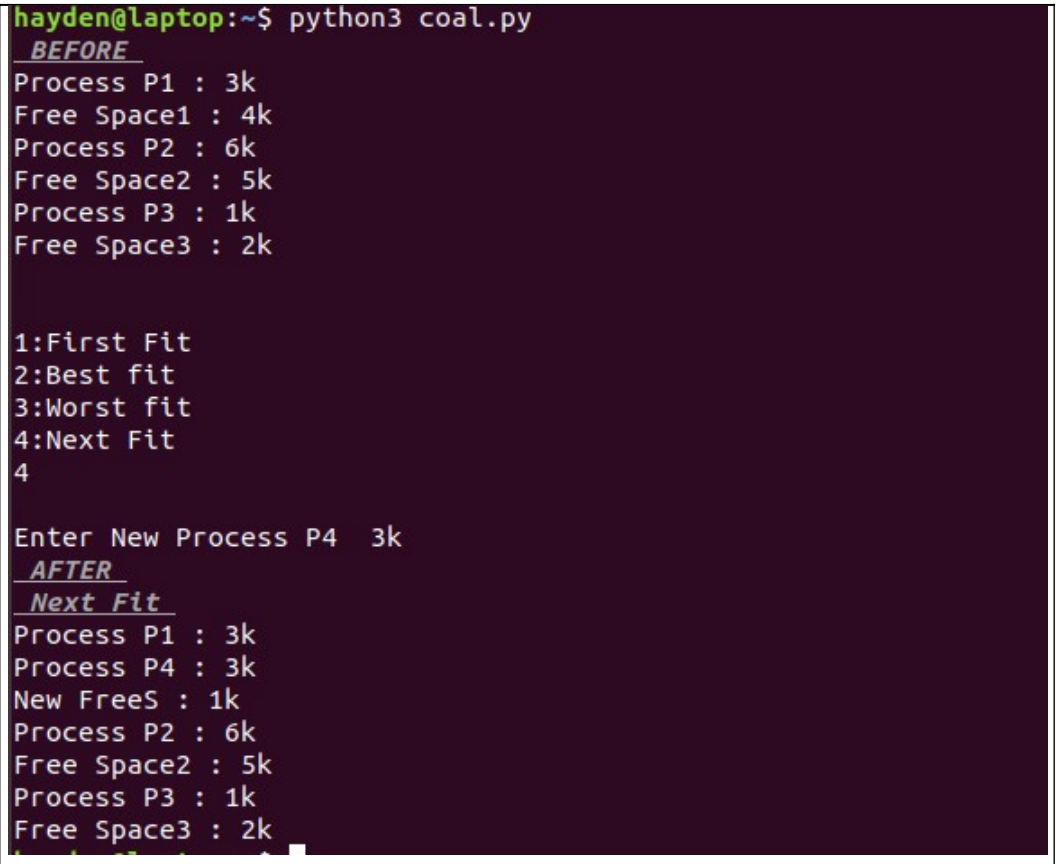
```
hayden@laptop:~$ python3 coal.py
 BEFORE
Process P1 : 3k
Free Space1 : 4k
Process P2 : 6k
Free Space2 : 5k
Process P3 : 1k
Free Space3 : 2k


1:First Fit
2:Best fit
3:Worst fit
4:Next Fit
2

Enter New Process P4  2k
 AFTER
 Best Fit
Process P1 : 3k
Free Space1 : 4k
Process P2 : 6k
Free Space2 : 5k
Process P3 : 1k
Process P4 : 2k
```

Worst Fit(New Process 2K)

```
hayden@laptop:~$ python3 coal.py
 BEFORE
Process P1 : 3k
Free Space1 : 4k
Process P2 : 6k
Free Space2 : 5k
Process P3 : 1k
Free Space3 : 2k

 Sublime Text
1:First Fit
2:Best fit
3:Worst fit
4:Next Fit
3

Enter New Process P4  2k
 AFTER
 Worst Fit
Process P1 : 3k
Free Space1 : 4k
Process P2 : 6k
Process P4 : 2k
New FreeS : 3k
Process P3 : 1k
Free Space3 : 2k
```

Next

	Fit(New Process 2K)
	
	`hayden@laptop:~$ python3 coal.py`
	` BEFORE `
	`Process P1 : 3k`
	`Free Space1 : 4k`
	`Process P2 : 6k`
	`Free Space2 : 5k`
	`Process P3 : 1k`
	`Free Space3 : 2k`
	
	`1:First Fit`
	`2:Best fit`
	`3:Worst fit`
	`4:Next Fit`
	`4`
	
	`Enter New Process P4  2k`
	` AFTER `
	` Next Fit `
	`Process P1 : 3k`
	`Free Space1 : 4k`
	`Process P2 : 6k`
	`Free Space2 : 5k`
	`Process P3 : 1k`
	`Process P4 : 2k`
	
	Next Fit(New Process 3K)

	
CONCLUSION	We have successfully understood and implemented Best Fit ,Next-Fit ,Worst-Fit,Next Algorithms
REFERENCES	1. William Stallings, "Computer Organization and Architecture: Designing for Performance", Pearson Publication, 10th Edition, 2013 2. B. Govindarajulu, "Computer Architecture and Organization: Design Principles and Applications", Second Edition, McGraw Hill (India)