F :. Conceicao Rodrigues College of Engineering Department of Computer Engineering			
Student's Roll No	**9536**	**Students Name**	**Saville Dsilva**
Date of Performance		**SE Computer – Div**	**A**

Aim: Study Memory Management **Lab Outcome:**

CSL403.4: Implement various memory management techniques and evaluate their performances.

Problem Statements:
Implement Dynamic Partitioning Placement Algorithms

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

1. Given the number of holes and their sizes, number of blocks to be placed in memory and their sizes, find which algorithm would be resulting in effective utilization of memory. 2. Give the allotment of blocks to holes in each algorithm **References:**
https://www.youtube.com/watch?v=oYfzZU2Z6Tk&t=626s

On time Submission(2)	**Knowledge of Topic(4)**	**Implementation and Demonstraion(4)**	**Total (10)**
Signature of Faculty		**Date of Submission**	

```
In [8]: num_holes=int(input("Enter the number of holes:")) holes=[] for i in range(num_holes):
size=int(input("enter the size of holes".format(i+1))) holes.append(size)

num_block=int(input("enter the number to processes:")) blocks=[] for i in range(num_block):
size=int(input("Enter the size of process{}:".format(i+1))) blocks.append(size)
holes.sort() print("the holes available for allocation of 1 process is:",holes) allocation=[-
1]*num_block for i in range(num_block): best_index=-1 for j in range(num_holes): if
holes[j]>=blocks[i]:
                        if best_index==-1 or holes[j]<holes[best_index]:
                                best_index=j
        if best_index!=-1:
                allocation[i]=best_index holes[best_index]-=blocks[i]
                print("the holes available for allocation of {} process is :{ print("process \t size \t
allocation") for i in range(num_block): print("{} \t\t {} \t\t{}".format(i+1,blocks[i],allocation[i]+1 if
```

```
Enter the number of holes:5 enter the size of
holes100 enter the size of holes300 enter the
size of holes200 enter the size of holes600
enter the size of holes400 enter the number to
processes:5 Enter the size of process1:212
Enter the size of process2:314
Enter the size of process3:112
Enter the size of process4:50 Enter the size of process5:20
the holes available for allocation of 1 process is: [100, 200, 300,
400, 600]
the holes available for allocation of 2 process is :[100, 200, 88,
400, 600]
the holes available for allocation of 3 process is :[100, 200, 88,
86, 600]
the holes available for allocation of 4 process is :[100, 88, 88, 8
6, 600]
the holes available for allocation of 5 process is :[100, 88, 88, 3
6, 600]
the holes available for allocation of 6 process is :[100, 88, 88, 1
6, 600]
  process                         size  allocation
1                               212  3
2                               314  4
3                               112  2
4                               50  4
5                               20  4
```

In

```
public class worstfit {

 public static void main(String[] args){

    int[] prSize = {200, 150, 100, 50, 300};

    int[] holeSize = {200, 300, 100, 150, 50};

    performWF(prSize, holeSize);


 }

 public static void performWF(int[] prSize, int [] holeSize){
    int max = findMax(holeSize);

    for(int i = 0; i<prSize.length; i++){
       if(holeSize[max]>= prSize[i]){
          System.out.println("Process index:" +i +" alloted");
          holeSize[max] = holeSize[max] - prSize[i];

       }
       else{
          System.out.println("Process index:" +i +" cannot be alloted");
       }

       max = findMax(holeSize);
    }


 }

 public static int findMax(int[] holeSize){
    int max = 0;

    for(int i = 0; i< holeSize.length; i++){
       if(holeSize[max] <= holeSize[i]){
          max = i;
       }
    }

    return max;

 }

}


 /* Output:
 Process index:0 alloted
Process index:1 alloted
Process index:2 alloted
Process index:3 alloted
Process index:4 cannot be alloted
*/
```

In [25]:
```
def firstFit(blockSize ,noBlocks,processSize,noProcess):
    allocation =[-1]*noProcess
    for i in range(noProcess):
        for j in range(noBlocks):
            if blockSize[j] >= processSize[i]:
                allocation[i]=j
                blockSize[j]-=processSize[i]
                break
    print("process no \t process size \t block size")
    for i in range(noProcess):
        print("     ",i+1,"           ",processSize[i],"       ", enc
        if allocation[i] != -1:
            print(allocation[i]+1)
        else:
            print("not allocated")


blockSize=[100,300,200,600,400]
processSize=[50,212,112,314]
noBlocks = len(blockSize)
noProcess = len(processSize)
firstFit(blockSize,noBlocks,processSize,noProcess)
```

```
process no      process size    block size
      1             50             1
      2             212             2
      3             112             3
      4             314             4
```

In [ ]:

In [ ]:

In [ ]: