Fr. Conceicao Rodrigues College of Engineering Department of Computer Engineering			
Student's Roll No	**9913**	**Students Name**	**Mark Lopes**
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

a) **best fit**

```
class PlacedBlock:

    def __init__(self, hole_size, block_size):

        self.hole_size = hole_size

        self.block_size = block_size


    def remaining_space(self):

        return self.hole_size - self.block_size



def best_fit_allocation(holes, blocks):

    placements = []


    for block in blocks:
```

```
        best_fit_index = -1

        # Find the best fit hole for the current block
        for i, hole in enumerate(holes):
            if hole >= block:
                if best_fit_index == -1 or hole < holes[best_fit_index]:
                    best_fit_index = i

        if best_fit_index != -1:
            # Allocate the block to the best fit hole
            allocated_hole_size = holes[best_fit_index]
            placements.append(PlacedBlock(allocated_hole_size, block))
            holes[best_fit_index] -= block  # Update the hole size after
allocation

    # Calculate total remaining space in all holes after allocation
    total_remaining_space = sum(hole for hole in holes)
    return placements, total_remaining_space



def main():
    # Initial setup
    holes = [100, 125, 150, 80, 50, 90]
    blocks = [90, 70, 140, 120, 20, 10]

    # Perform Best Fit allocation
    allocated_blocks, remaining_space = best_fit_allocation(holes, blocks)

    # Display results
```

```
    print("Allocated Blocks:")

    for placement in allocated_blocks:

        print(f"Block Size: {placement.block_size}, Hole Size After
Allocation: {placement.remaining_space()}")



    print(f"\nTotal Remaining Space in Holes: {remaining_space}")




if __name__ == "__main__":

    main()
```

```
Allocated Blocks:
Block Size: 90, Hole Size After Allocation: 0
Block Size: 70, Hole Size After Allocation: 10
Block Size: 140, Hole Size After Allocation: 10
Block Size: 120, Hole Size After Allocation: 5
Block Size: 20, Hole Size After Allocation: 30
Block Size: 10, Hole Size After Allocation: 0

Total Remaining Space in Holes: 145
PS C:\Users\Mark Lopes\Desktop\college\Sem_4\Os>
```

b) first-fit

```
class PlacedBlock:

    def __init__(self, hole_size, block_size):

        self.hole_size = hole_size

        self.block_size = block_size


    def remaining_space(self):

        return self.hole_size - self.block_size
```

```
def first_fit_allocation(holes, blocks):
    placements = []

    for block in blocks:
        allocated = False

        # Iterate through each hole to find the first fit
        for i, hole in enumerate(holes):
            if hole >= block:
                # Allocate the block to the current hole
                placements.append(PlacedBlock(hole, block))
                holes[i] -= block  # Update the hole size after
allocation
                allocated = True
                break  # Stop searching for holes after first fit

        if not allocated:
            # If no hole can accommodate the block, mark as not
allocated
            placements.append(PlacedBlock(0, block))  # Hole size of 0
indicates not allocated

    # Calculate total remaining space in all holes after allocation
    total_remaining_space = sum(holes)
    return placements, total_remaining_space


def main():
    holes = [100, 125, 150, 80, 50, 90]
```

```
    blocks = [90, 70, 140, 120, 20, 10]


    # Perform First Fit allocation
    allocated_blocks, remaining_space = first_fit_allocation(holes[:],
blocks)


    # Display results
    print("Allocated Blocks:")
    for placement in allocated_blocks:
        if placement.hole_size > 0:
            print(f"Block Size: {placement.block_size}, Hole Size
After Allocation: {placement.remaining_space()}")
        else:
            print(f"Block Size: {placement.block_size}, Not
Allocated")


    print(f"\nTotal Remaining Space in Holes: {remaining_space}")



if __name__ == "__main__":
    main()
```

```
PS C:\Users\Mark Lopes\Desktop\college\Sem_4\C
Allocated Blocks:
Block Size: 90, Hole Size After Allocation: 10
Block Size: 70, Hole Size After Allocation: 55
Block Size: 140, Hole Size After Allocation: 1
Block Size: 120, Not Allocated
Block Size: 20, Hole Size After Allocation: 35
Block Size: 10, Hole Size After Allocation: 0

Total Remaining Space in Holes: 265
```

c) Worst-Fit

```
class PlacedBlock:

    def __init__(self, hole_size, block_size):

        self.hole_size = hole_size

        self.block_size = block_size


    def remaining_space(self):

        return self.hole_size - self.block_size




def worst_fit_allocation(holes, blocks):

    placements = []


    for block in blocks:

        worst_fit_index = -1


        # Find the largest hole that can accommodate the block

        for i, hole in enumerate(holes):

            if hole >= block:

                if worst_fit_index == -1 or hole >
holes[worst_fit_index]:

                    worst_fit_index = i
```

```
        if worst_fit_index != -1:
            # Allocate the block to the largest hole found
            placements.append(PlacedBlock(holes[worst_fit_index], block))
            holes[worst_fit_index] -= block  # Update the hole size after allocation

    # Calculate total remaining space in all holes after allocation
    total_remaining_space = sum(holes)
    return placements, total_remaining_space


def main():
    holes = [100, 125, 150, 80, 50, 90]
    blocks = [90, 70, 140, 120, 20, 10]

    # Perform Worst Fit allocation
    allocated_blocks, remaining_space = worst_fit_allocation(holes[:], blocks)

    # Display results
    print("Allocated Blocks:")
    for placement in allocated_blocks:
        print(f"Block Size: {placement.block_size}, Hole Size After Allocation: {placement.remaining_space()}")

    print(f"\nTotal Remaining Space in Holes: {remaining_space}")
```

```
if __name__ == "__main__":
    main()
```

```
PS C:\Users\Mark Lopes\Desktop\college\Sem_4\Os> pyt
Allocated Blocks:
Block Size: 90, Hole Size After Allocation: 60
Block Size: 70, Hole Size After Allocation: 55
Block Size: 20, Hole Size After Allocation: 80
Block Size: 10, Hole Size After Allocation: 80

Total Remaining Space in Holes: 405
PS C:\Users\Mark Lopes\Desktop\college\Sem_4\Os>
```

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