Name : Devesh Mali  
Roll No :13228  
Batch : B2  
  
 Practical 6

class FirstFit:

    def firstFit(self, blockSize, processSize):

        m = len(blockSize)

        n = len(processSize)

        allocation = [-1] \* n

        for i in range(n):

            for j in range(m):

                if blockSize[j] >= processSize[i]:

                    allocation[i] = j

                    blockSize[j] -= processSize[i]

                    break

        print("\nProcess No.\tProcess Size\tBlock no.")

        for i in range(n):

            print(f"{i + 1}\t\t{processSize[i]}\t\t", end="")

            if allocation[i] != -1:

                print(allocation[i] + 1)

            else:

                print("Not Allocated")

class NextFit:

    def nextFit(self, blockSize, processSize):

        m = len(blockSize)

        n = len(processSize)

        allocation = [-1] \* n

        j = 0

        for i in range(n):

            count = 0

            while count < m:

                if blockSize[j] >= processSize[i]:

                    allocation[i] = j

                    blockSize[j] -= processSize[i]

                    break

                j = (j + 1) % m

                count += 1

        print("\nProcess No.\tProcess Size\tBlock no.")

        for i in range(n):

            print(f"{i + 1}\t\t{processSize[i]}\t\t", end="")

            if allocation[i] != -1:

                print(allocation[i] + 1)

            else:

                print("Not Allocated")

class WorstFit:

    def worstFit(self, blockSize, processSize):

        m = len(blockSize)

        n = len(processSize)

        allocation = [-1] \* n

        for i in range(n):

            wstIdx = -1

            for j in range(m):

                if blockSize[j] >= processSize[i]:

                    if wstIdx == -1 or blockSize[wstIdx] < blockSize[j]:

                        wstIdx = j

            if wstIdx != -1:

                allocation[i] = wstIdx

                blockSize[wstIdx] -= processSize[i]

        print("\nProcess No.\tProcess Size\tBlock no.")

        for i in range(n):

            print(f"{i + 1}\t\t{processSize[i]}\t\t", end="")

            if allocation[i] != -1:

                print(allocation[i] + 1)

            else:

                print("Not Allocated")

class BestFit:

    def bestFit(self, blockSize, processSize):

        m = len(blockSize)

        n = len(processSize)

        allocation = [-1] \* n

        for i in range(n):

            bestIdx = -1

            for j in range(m):

                if blockSize[j] >= processSize[i]:

                    if bestIdx == -1 or blockSize[bestIdx] > blockSize[j]:

                        bestIdx = j

            if bestIdx != -1:

                allocation[i] = bestIdx

                blockSize[bestIdx] -= processSize[i]

        print("\nProcess No.\tProcess Size\tBlock no.")

        for i in range(n):

            print(f"{i + 1}\t\t{processSize[i]}\t\t", end="")

            if allocation[i] != -1:

                print(allocation[i] + 1)

            else:

                print("Not Allocated")

def main():

    first = FirstFit()

    next\_fit = NextFit()

    worst = WorstFit()

    best = BestFit()

    while True:

        print("\nEnter the number of Blocks: ")

        m = int(input())

        print("Enter the number of Processes: ")

        n = int(input())

        blockSize = list(map(int, input("Enter the Size of all the blocks (space-separated): ").split()))

        processSize = list(map(int, input("Enter the Size of all the processes (space-separated): ").split()))

        print("\nMenu")

        print("1. First Fit")

        print("2. Next Fit")

        print("3. Worst Fit")

        print("4. Best Fit")

        print("5. Exit")

        choice = int(input("Select the algorithm you want to implement: "))

        if choice == 1:

            print("First Fit Output")

            first.firstFit(blockSize[:], processSize)  # Pass a copy to avoid modifying the original list

        elif choice == 2:

            print("Next Fit Output")

            next\_fit.nextFit(blockSize[:], processSize)

        elif choice == 3:

            print("Worst Fit Output")

            worst.worstFit(blockSize[:], processSize)

        elif choice == 4:

            print("Best Fit Output")

            best.bestFit(blockSize[:], processSize)

        elif choice == 5:

            print("Exiting the code...")

            break

        else:

            print("Invalid option")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

OUTPUT :

PS C:\Users\HP> & "C:/Program Files/Python312/python.exe" "c:/Users/HP/OneDrive/Desktop/spos practical/lexical/fifo.py"

Enter the number of Blocks:

5

Enter the number of Processes:

4

Enter the Size of all the blocks (space-separated): 100 500 200 300 600

Enter the Size of all the processes (space-separated): 212 417 112 426

Menu

1. First Fit

2. Next Fit

3. Worst Fit

4. Best Fit

5. Exit

Select the algorithm you want to implement: 1

First Fit Output

Process No. Process Size Block no.

1 212 2

2 417 5

3 112 3

4 426 Not Allocated

Enter the number of Blocks:

5

Enter the number of Processes:

4

Enter the Size of all the blocks (space-separated): 100 500 200 300 600

Enter the Size of all the processes (space-separated): 212 417 116 426

Menu

1. First Fit

2. Next Fit

3. Worst Fit

4. Best Fit

5. Exit

Select the algorithm you want to implement: 2  
Next Fit Output   
  
  
Process No. Process Size Block no.

1 212 2

2 417 5

3 112 1

4 426 Not Allocated

Enter the number of Blocks:

5

Enter the number of Processes:

4

Enter the Size of all the blocks (space-separated): 100 500 200 300 600

Enter the Size of all the processes (space-separated): 212 417 112 426

Menu

1. First Fit

2. Next Fit

3. Worst Fit

4. Best Fit

5. Exit

Select the algorithm you want to implement: 3

Worst Fit Output

Process No. Process Size Block no.

1 212 5

2 417 2

3 112 4

4 426 Not Allocated

Enter the number of Blocks:

5

Enter the number of Processes:

4

Enter the Size of all the blocks (space-separated): 100 500 200 300 600

Enter the Size of all the processes (space-separated): 212 417 112 426

Menu

1. First Fit

2. Next Fit

3. Worst Fit

4. Best Fit

5. Exit

Select the algorithm you want to implement: 4

Best Fit Output

Process No. Process Size Block no.

1 212 4

2 417 2

3 112 3

4 426 5

Conclusion :

The memory allocation algorithms demonstrate varied efficiency in allocating blocks to processes.

* **First Fit** allocated 3 processes successfully, leaving one unallocated.
* **Next Fit** allocated 3 processes as well, but utilized a different block for the third process.
* **Worst Fit** allocated 2 processes to the largest blocks, leaving one unallocated.
* **Best Fit** achieved optimal allocation by fitting all processes into the smallest available blocks, successfully allocating each one.

Overall, **Best Fit** yielded the most efficient memory usage.