```python
#!/usr/bin/env python3

from __future__ import print_function
import sys

'''
Course: CSC 314
Description: Implementaion of best fit & first fit memory allocation algorithm
Group: 5
Members:
    1. Mumuni Abdullah          180805047
    2. Olorunfemi-Ojo Tomiwa    190805503
    3. Animashaun Sofiat        190805520
    4. Ogunrinde Motunrayo      190805514
    5. Ademuyiwa Abdulbaqi      190805522
'''

def set_parameter_values():
    '''
    Sets parameters needed by the algorithm
        Parameters:
            MEMORY_BLOCK_SIZE     (int): The size of the memory block
            NUMBER_OF_PARTITIONS  (int): The number of partitions in memory
            PARTITION_SIZES      (list): The sizes of the partitions in memory
            PROCESS_SIZES        (list): The sizes of the jobs to be allocated memory
    '''
    global MEMORY_BLOCK_SIZE, NUMBER_OF_PARTITIONS, PARTITION_SIZES, PROCESS_SIZES
    try:
        MEMORY_BLOCK_SIZE = int(input("[+] Enter total size of memory block (KB): "))
        NUMBER_OF_PARTITIONS = int(input("[+] Enter the number of partitions: "))
        PARTITION_SIZES = [int(x) for x in input("[+] Enter the partition sizes seperated by
commas (KB): ").split(",")]
        if len(PARTITION_SIZES) !=  NUMBER_OF_PARTITIONS:
            print(f"[+] Error: expected {NUMBER_OF_PARTITIONS} partition sizes")
            sys.exit(1)
        elif sum(PARTITION_SIZES) != MEMORY_BLOCK_SIZE:
            print(f"[+] Error: sum of partition sizes ({sum(PARTITION_SIZES)}KB) is not equal
to memory size({MEMORY_BLOCK_SIZE}KB)")
            sys.exit(1)
        PROCESS_SIZES = [int(x) for x in input("[+] Enter the process sizes seperated by
commas (KB): ").split(",")]
        if len(PROCESS_SIZES) !=  NUMBER_OF_PARTITIONS:
            print(f"[+] Error: expected {NUMBER_ONUMBER_OF_PARTITIONS} process sizes")
            sys.exit(1)
    except KeyboardInterrupt:
```

```
        print("\n[+] Program was abruptly terminated")
        sys.exit(1)
    except ValueError:
        print("[+] Error: expected an integer")
        sys.exit(1)


def display_parameter_values():
    '''Displays the inputed parameters'''
    print(f"\n[+] Memory Size: {MEMORY_BLOCK_SIZE}KB")
    print(f"[+] Partition Sizes (KB): {str(PARTITION_SIZES)[1:-1]}\n")
    print(f"Process List:\n{'-'*30}")
    print("| {:<11} {:<16}".format("Process No.", "Process Size(KB)"))
    print(f"|{'-'*28}")
    for i, process_size in enumerate(PROCESS_SIZES):
        print("| {:<11} {:<16}".format(f"P{i + 1}", str(process_size)))
    print(f"{'-'*30}\n")


def display_output(title, output, total_used_memory, total_fragment_size,
unallocated_processes):
    '''
    Tabulates the output
        args:
            title                 (str): The name of the algorithm
            output               (dict): Key    (int) = Partition Number
                                         Value (list) = [Block Size, Process Number, Process
Size, Process Status, Fragment Size]
            total_used_memory     (int): Sum of utilized memory
            total_fragment_size   (int): Sum of fragmented memory
            unallocated_processes (list): List of jobs without memory allocation
    '''
    print(f"{title}:\n{'-'*90}")
    print(
        "| {:<14} {:<15} {:<12} {:<17} {:<7} {:<18}"
        .format("Partition No.", "Block Size(KB)", "Process No.", "Process Size(KB)",
"Status", "Fragment Size(KB)")
    )
    print(f"|{'-'*89}")
    for key in sorted(output):
        print(
            "| {:<14} {:<15} {:<12} {:<17} {:<7} {:<18}"
            .format(key, str(output[key][0]), output[key][1], str(output[key][2]),
output[key][3], str(output[key][4]))
        )
```

```
    print(f"{'-'*90}")
    print(f"[+] Total memory used: {total_used_memory}KB")
    print(f"[+] Total fragment size: {total_fragment_size}KB")
    print(f"[+] Processes without allocated memory: {str(unallocated_processes)[1:-1]}\n")


def best_fit():
    '''
    Best fit algorithm implementation
        parameters:
            output                     (dict): Key    (int) = Partition Number
                                               Value (list) = [Block Size, Process Number,
Process Size, Process Status, Fragment Size]
            is_partition_taken         (list): Tracks if a partition has been assigned a job
            unallocated_processes      (list): List of jobs without memory allocation
            total_used_memory           (int): Sum of utilized memory
            total_fragment_size         (int): Sum of fragmented memory
            potential_occupiable_space (dict): Memory locations greater than or equal to the
job size
                                               Key    (int) = Partition Number
                                               Value  (int) = Partition Size
    '''
    output, is_partition_taken, unallocated_processes = {}, [False] * NUMBER_OF_PARTITIONS, []
    total_used_memory, total_fragment_size = 0, 0
    for i, process_size in enumerate(PROCESS_SIZES):
        potential_occupiable_space = {}
        for j, partition_size in enumerate(PARTITION_SIZES):
            if is_partition_taken[j] == True:
                continue
            elif process_size <= partition_size:
                potential_occupiable_space[j + 1] = partition_size
            else:
                output[j + 1] = [partition_size, "-", "-", "Free", "-"]
        if len(potential_occupiable_space) != 0:
            smallest_partition = min(potential_occupiable_space.values())
            smallest_partition_index = min(potential_occupiable_space,
key=potential_occupiable_space.get)
            output[smallest_partition_index] = [smallest_partition, f"P{i + 1}", process_size,
"Busy", (smallest_partition - process_size)]
            is_partition_taken[smallest_partition_index - 1] = True
            total_used_memory = total_used_memory + process_size
            total_fragment_size = total_fragment_size + (smallest_partition - process_size)
        else:
            unallocated_processes.append(f"P{i + 1}")
```

```
    display_output("Best Fit Method", output, total_used_memory, total_fragment_size,
unallocated_processes)


def first_fit():
    '''
    First fit algorithm implementation
        parameters:
            output                (dict): Key    (int) = Partition Number
                                          Value (list) = [Block Size, Process Number, Process
Size, Process Status, Fragment Size]
            is_partition_taken    (list): Tracks if a partition has been assigned a job
            is_process_x_taken    (dict): Tracks if a job has been assigned a memory location
            unallocated_processes (list): List of jobs without memory allocation
            total_used_memory      (int): Sum of utilized memory
            total_fragment_size    (int): Sum of fragmented memory
    '''
    output, is_partition_taken, unallocated_processes = {}, [False] * NUMBER_OF_PARTITIONS, []
    is_process_x_taken = {}
    total_used_memory, total_fragment_size = 0, 0
    for i, process_size in enumerate(PROCESS_SIZES):
        for j, partition_size in enumerate(PARTITION_SIZES):
            if is_partition_taken[j] == True:
                continue
            elif process_size <= partition_size:
                output[j + 1] = [partition_size, f"P{i + 1}", process_size, "Busy",
(partition_size - process_size)]
                total_used_memory = total_used_memory + process_size
                total_fragment_size = total_fragment_size + (partition_size - process_size)
                is_partition_taken[j], is_process_x_taken[i + 1] = True, True
                break
            else:
                output[j + 1] = [partition_size, "-", "-", "Free", "-"]
                is_process_x_taken[i + 1] = False
    for key in is_process_x_taken:
        if is_process_x_taken[key] == False:
            unallocated_processes.append(f"P{key}")
    display_output("First Fit Method", output, total_used_memory, total_fragment_size,
unallocated_processes)


def main():
    set_parameter_values()
    display_parameter_values()
    first_fit()
```

```
    best_fit()


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

OUTPUT:

```
[+] Enter total size of memory block (KB): 1000
[+] Enter the number of partitions: 4
[+] Enter the partition sizes seperated by commas (KB): 300, 400, 100, 200
[+] Enter the process sizes seperated by commas (KB): 200, 450, 50, 300

[+] Memory Size: 1000KB
[+] Partition Sizes (KB): 300, 400, 100, 200

Process List:
------------------------------
| Process No. Process Size(KB)
|-----------------------------
| P1          200
| P2          450
| P3          50
| P4          300
------------------------------

First Fit Method:
-----------------------------------------------------------------------------------------
| Partition No.  Block Size(KB)  Process No.  Process Size(KB)  Status  Fragment Size(KB)
|----------------------------------------------------------------------------------------
| 1              300             P1           200              Busy    100
| 2              400             P3           50               Busy    350
| 3              100             -            -                Free    -
| 4              200             -            -                Free    -
-----------------------------------------------------------------------------------------
[+] Total memory used: 250KB
[+] Total fragment size: 450KB
[+] Processes without allocated memory: 'P2', 'P4'

Best Fit Method:
-----------------------------------------------------------------------------------------
| Partition No.  Block Size(KB)  Process No.  Process Size(KB)  Status  Fragment Size(KB)
|----------------------------------------------------------------------------------------
| 1              300             P4           300              Busy    0
| 2              400             -            -                Free    -
| 3              100             P3           50               Busy    50
| 4              200             P1           200              Busy    0
-----------------------------------------------------------------------------------------
[+] Total memory used: 550KB
[+] Total fragment size: 50KB
[+] Processes without allocated memory: 'P2'
```

DOCUMENTATION

Best Fit and First Fit are algorithms that allocate memory to processes. The First Fit algorithm allocates the first memory partition that is large enough to contain that process while the Best Fit algorithm looks for the smallest partition that is large enough to contain the process. For the implementation of these algorithms, some parameters are required:

1. `MEMORY_BLOCK_SIZE`: size of the block of memory to be partitioned and allocated to processes.
2. `NUMBER_OF_PARTTIONS`: number of segments the memory block is divided into.
3. `PARTITION_SIZES`: list of the sizes of the memory partitions. The sum of these values must equal `MEMORY_BLOCK_SIZE`.
4. `PROCESS_SIZES`: list of the sizes of the processes.

HOW DO THEY WORK?

In the First Fit algorithm, the size of every process is compared to the sizes of all unclaimed partitions. The first partition that is greater than or equal to the size of the process is assigned.

```
for i, process_size in enumerate(PROCESS_SIZES):
    for j, partition_size in enumerate(PARTITION_SIZES):
        if is_partition_taken[j] == True:
            continue  # checks if the partition has been claimed by another process
        elif process_size <= partition_size:
            # assign this partition to the process
            break
```

Considering the code output in page 5, a system has a memory block of size 1000KB. It is divided into 4 partitions of P1 (300KB), P2 (400KB), P3 (100KB) and P4 (200KB). The system has 4 jobs to perform, initialize a browsing session (J1 requires 200KB), start a video game (J2 - 450KB), play music (J3 - 50KB) and load a Word document (J4 - 300KB), The First Fit algorithm allocates J1 (200KB) to P1 (300KB) because it is the first partition large enough to initialize the browsing session. It wastes 100KB since it only requires 200KB. J2 i.e. the video game will not be started because there is no partition large enough among the free partitions to load the game. J3 is assigned to P2 (400KB). 350KB is wasted. P3 and P4 are not allocated any jobs. Out of the 1000KB, 250KB is used to perform 2 jobs, 450KB is wasted and 300KB is not allocated any job.

For the Best Fit algorithm, the size of every process is compared to the sizes of all unclaimed partitions. The smallest partition from the list of partitions that are greater than or equal to the process size is chosen.

```
for i, process_size in enumerate(PROCESS_SIZES):
    potential_occupiable_space = {} # partitions large enough to contain the process
    for j, partition_size in enumerate(PARTITION_SIZES):
        if is_partition_taken[j] == True:
            continue  # checks if the partition has been claimed by another process
        elif process_size <= partition_size:
            # add partition to potential occupiable partitions
```

```
            potential_occupiable_space[j + 1] = partition_size
    if len(potential_occupiable_space) != 0:
        # find the smallest partition
        smallest_partition = min(potential_occupiable_space.values())
        # assign this smallest_partition to the process
```

Using the same example in First Fit, the browsing session (J1 – 200KB) can be assigned to P1 (300KB), P2 (400KB) or P4 (200KB). P4 is chosen because it is the smallest that can accommodate J1. The video game is not loaded because no partition can accommodate it. The music player is loaded into P3 (100KB) (50KB wasted) over P2 (400KB) since P3 is smaller and lastly, the Word document is assigned to P1. P2 is not assigned a job. Out of the 1000KB, 550KB is used to perform 3 jobs, 50KB is wasted and 400KB is not used.