4. Write a C/Python program to calculate average waiting time and Turnaround Time of n processes with First Come First Serve (FCFS) CPU scheduling algorithm.

```
bt = [5, 3, 8]  # burst times
n = len(bt)
wt = [0]*n
tat = [0]*n

for i in range(1, n):
    wt[i] = wt[i-1] + bt[i-1]
for i in range(n):
    tat[i] = wt[i] + bt[i]

print("Waiting Times:", wt)
print("Turnaround Times:", tat)
print("Average Waiting Time:", sum(wt)/n)
print("Average Turnaround Time:", sum(tat)/n)
```

```
Waiting Times: [0, 5, 8]
Turnaround Times: [5, 8, 16]
Average Waiting Time: 4.333333333333333
Average Turnaround Time: 9.666666666666666
```

5.Write a C/Python program to calculate average waiting time and Turnaround Time of n processes with Shortest Job First (SJF) CPU scheduling algorithm.(non preemptive)

```
bt = [6, 8, 7, 3]  # burst times
n = len(bt)
p = list(range(n))
p.sort(key=lambda x: bt[x])

wt = [0]*n
tat = [0]*n
for i in range(1, n):
    wt[p[i]] = wt[p[i-1]] + bt[p[i-1]]
for i in range(n):
    tat[i] = wt[i] + bt[i]

print("Waiting Times:", wt)
print("Turnaround Times:", tat)
print("Average Waiting Time:", sum(wt)/n)
print("Average Turnaround Time:", sum(tat)/n)
```

```
Waiting Times: [3, 16, 9, 0]
Turnaround Times: [9, 24, 16, 3]
Average Waiting Time: 7.0
Average Turnaround Time: 13.0
```

6.Write a C/Python program to calculate average waiting time and Turnaround Time of n processes with Priority CPU scheduling algorithm.(non preemptive)

```
bt = [10, 1, 2, 1]          # burst times
priority = [3, 1, 4, 2]     # lower number = higher priority
n = len(bt)
p = list(range(n))
p.sort(key=lambda x: priority[x])

wt = [0]*n
tat = [0]*n
for i in range(1, n):
    wt[p[i]] = wt[p[i-1]] + bt[p[i-1]]
for i in range(n):
    tat[i] = wt[i] + bt[i]

print("Waiting Times:", wt)
print("Turnaround Times:", tat)
print("Average Waiting Time:", sum(wt)/n)
print("Average Turnaround Time:", sum(tat)/n)
```

```
Waiting Times: [2, 0, 12, 1]
Turnaround Times: [12, 1, 14, 2]
Average Waiting Time: 3.75
Average Turnaround Time: 7.25
```

7.Write a C/Python program to calculate average waiting time and Turnaround Time of n processes with Round Robin (RR) CPU

```
bt = [5, 4, 2]  # burst times
q = 2           # time quantum
n = len(bt)
rt = bt.copy()
t = 0
wt = [0]*n

while max(rt) > 0:
    for i in range(n):
        if rt[i] > 0:
            dt = min(rt[i], q)
            t += dt
            rt[i] -= dt
            if rt[i] == 0:
                wt[i] = t - bt[i]

tat = [wt[i] + bt[i] for i in range(n)]
print("Waiting Times:", wt)
print("Turnaround Times:", tat)
print("Average Waiting Time:", sum(wt)/n)
print("Average Turnaround Time:", sum(tat)/n)
```

```
Waiting Times: [6, 6, 4]
Turnaround Times: [11, 10, 6]
Average Waiting Time: 5.333333333333333
Average Turnaround Time: 9.0
```

10.Write a C/Python program on First In First Out (FIFO) Page Replacement algorithm.

```
pages = [1, 2, 3, 2, 4, 1, 5]
frames = 3
memory = []
faults = 0
hits = 0
for p in pages:
    if p not in memory:
        if len(memory) < frames:
            memory.append(p)
        else:
            memory.pop(0)
            memory.append(p)
        faults += 1
    else:
        hits += 1

print("Page Faults:", faults)
print("Page Hits:", hits)
print("Hit Ratio:", hits / len(pages))
```

```
Page Faults: 6
Page Hits: 1
Hit Ratio: 0.14285714285714285
```

11.Write a C/Python program on Least Recently Used (LRU) Page Replacement algorithm.

```
pages = [1, 2, 3, 2, 2, 5, 1]
frames = 3
memory = []
faults = 0
hits = 0

for p in pages:
    if p in memory:
        # Page hit
        hits += 1
        memory.remove(p)
    else:
        # Page fault
        faults += 1
        if len(memory) == frames:
            memory.pop(0)
    memory.append(p)
print("Page Faults:", faults)
print("Page Hits:", hits)
print("Hit Ratio:", hits / len(pages))
```

```
Page Faults: 4
Page Hits: 3
Hit Ratio: 0.42857142857142855
```

12.Write a C/Python program on sequential file allocation method.

```
def sequential_file_allocation():
    n = int(input("Enter number of files: "))
    files = []

    for i in range(n):
        start = int(input(f"\nEnter starting block for File {i+1}: "))
        length = int(input(f"Enter length for File {i+1}: "))
        files.append([f"File {i+1}", start, length])

    print("\nFile\tStart Block\tLength\tBlocks Occupied")
    for name, start, length in files:
        blocks = list(range(start, start + length))
        print(f"{name}\t{start}\t\t{length}\t{blocks}")

sequential_file_allocation()
```

```
Enter number of files: 3

Enter starting block for File 1: 0
Enter length for File 1: 18

Enter starting block for File 2: 20
Enter length for File 2: 25

Enter starting block for File 3: 50
Enter length for File 3: 45

File    Start Block     Length  Blocks Occupied
File 1  0               18      [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]
File 2  20              25      [20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44]
File 3  50              45      [50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 8
3, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94]
```