Name: Bhargav Dhole
Roll No: 150096724014
Batch: Jensen Huang
Case Study No: 27

Problem Statement: Bank ATM Scheduling
Scenario: Customers arrive at an ATM randomly. VIP customers are prioritized to ensure faster access, but regular customers must also be served efficiently.

Proposed Solution:

1. First Come First Serve (FCFS)

Explanation:

Customers are served strictly in the order they arrive, regardless of priority.

Proposed Solution:

- Maintain a single queue sorted by arrival time.
- The ATM serves one customer completely before moving to the next.

2. Priority Scheduling (Preemptive)

Explanation:

VIP customers (priority = 0) are served before regular customers (priority = 1).

In the **preemptive version**, if a VIP arrives while a regular customer is being served, the regular customer's transaction is interrupted.

Proposed Solution:

- Use a priority queue.
- Always select the customer with the highest priority (lowest number).
- If a VIP arrives, pause the regular customer and resume them later.

3. Round Robin (Time-Sharing)

Explanation:

Each customer gets a fixed **time slice (quantum)** to use the ATM. If the transaction is not completed, the customer goes back to the end of the queue.

Proposed Solution:

- Maintain a circular queue of customers.

- Allocate a fixed time quantum to each customer.
- Repeat until all transactions finish.

Screenshots:

```c
#include <stdio.h>
#include <stdlib.h>

#define MAX_CUSTOMERS 50

typedef struct
{
    int customer_id;
    int arrival_time;
    int transaction_time;
    int priority;
    int remaining_time;
    int waiting_time;
    int turnaround_time;
} Customer;

void fcfs(Customer customers[], int n)
{
    int current_time = 0;
    float total_waiting_time = 0;
    float total_turnaround_time = 0;

    printf("\n--- FCFS SCHEDULING ---\n");
    printf("ID\tArrival\tBurst\tWaiting\tTurnaround\n");

    for (int i = 0; i < n; i++)
    {
        if (current_time < customers[i].arrival_time)
        {
            current_time = customers[i].arrival_time;
        }

        customers[i].waiting_time = current_time - customers[i].arrival_time;
        customers[i].turnaround_time =
            customers[i].waiting_time + customers[i].transaction_time;

        current_time += customers[i].transaction_time;

        total_waiting_time += customers[i].waiting_time;
        total_turnaround_time += customers[i].turnaround_time;

        printf("%d\t%d\t%d\t%d\t\t%d\n",
               customers[i].customer_id,
               customers[i].arrival_time,
               customers[i].transaction_time,
               customers[i].waiting_time,
               customers[i].turnaround_time);
    }

    printf("\nAverage Waiting Time: %.2f\n",
           total_waiting_time / n);
    printf("Average Turnaround Time: %.2f\n",
           total_turnaround_time / n);
}

void priority_preemptive(Customer customers[], int n)
{
    int time = 0, completed = 0;
    int highest_priority_index = -1;
    int min_priority;
    int is_completed[MAX_CUSTOMERS] = {0};

    float total_waiting_time = 0;
    float total_turnaround_time = 0;

    printf("\n--- PREEMPTIVE PRIORITY SCHEDULING (VIP FIRST) ---\n");

    while (completed < n)
    {
        min_priority = 999;
        highest_priority_index = -1;

        for (int i = 0; i < n; i++)
        {
            if (customers[i].arrival_time <= time &&
```

```
====== BANK ATM SCHEDULING SYSTEM ======
1. Enter Customer Details
2. FCFS Scheduling
3. Priority Scheduling (Preemptive)
4. Round Robin Scheduling
5. Exit
Enter your choice: 3

--- PREEMPTIVE PRIORITY SCHEDULING (VIP FIRST) ---
ID      Priority        Waiting Turnaround
1       1               0               1
2       1               0               2
3       0               0               3

Average Waiting Time: 0.00
Average Turnaround Time: 2.00

====== BANK ATM SCHEDULING SYSTEM ======
1. Enter Customer Details
2. FCFS Scheduling
3. Priority Scheduling (Preemptive)
4. Round Robin Scheduling
5. Exit
Enter your choice: 4
Enter Time Quantum: 1

--- ROUND ROBIN SCHEDULING ---
ID      Waiting Turnaround
1       0               1
2       0               2
3       0               3

Average Waiting Time: 0.00
Average Turnaround Time: 2.00

====== BANK ATM SCHEDULING SYSTEM ======
1. Enter Customer Details
2. FCFS Scheduling
3. Priority Scheduling (Preemptive)
4. Round Robin Scheduling
5. Exit
Enter your choice: 5
Exiting program.
bhargavdhole@Bhargavs-MacBook-Air ATM_Scheduling_OS %
```

```
            }
        }

        if (highest_priority_index == -1)
        {
            time++;
            continue;
        }

        customers[highest_priority_index].remaining_time--;
        time++;

        if (customers[highest_priority_index].remaining_time == 0)
        {
            is_completed[highest_priority_index] = 1;
            completed++;

            customers[highest_priority_index].turnaround_time =
                time - customers[highest_priority_index].arrival_time;

            customers[highest_priority_index].waiting_time =
                customers[highest_priority_index].turnaround_time -
                customers[highest_priority_index].transaction_time;

            total_waiting_time += customers[highest_priority_index].waiting_time;
            total_turnaround_time += customers[highest_priority_index].turnaround_time;
        }
    }

    printf("ID\tPriority\tWaiting\tTurnaround\n");
    for (int i = 0; i < n; i++)
    {
        printf("%d\t%d\t\t%d\t\t%d\n",
               customers[i].customer_id,
               customers[i].priority,
               customers[i].waiting_time,
               customers[i].turnaround_time);
    }

    printf("\nAverage Waiting Time: %.2f\n",
           total_waiting_time / n);
    printf("Average Turnaround Time: %.2f\n",
           total_turnaround_time / n);
}

Windsurf: Refactor | Explain | Generate Function Comment | ×
void round_robin(Customer customers[], int n, int time_quantum)
{
    int time = 0;
    int completed = 0;
    int queue[MAX_CUSTOMERS];
    int front = 0, rear = 0;

    int visited[MAX_CUSTOMERS] = {0};

    float total_waiting_time = 0;
    float total_turnaround_time = 0;

    // Initially add arrived customers at time 0
    for (int i = 0; i < n; i++)
    {
        if (customers[i].arrival_time == 0)
        {
            queue[rear++] = i;
            visited[i] = 1;
        }
    }

    printf("\n--- ROUND ROBIN SCHEDULING ---\n");

```

```
271
272         case 3:
273             for (int i = 0; i < n; i++)
274             {
275                 customers[i].remaining_time = customers[i].transaction_time;
276             }
277             priority_preemptive(customers, n);
278             break;
279
280         case 4:
281         {
282             int time_quantum;
283             printf("Enter Time Quantum: ");
284             scanf("%d", &time_quantum);
285
286             for (int i = 0; i < n; i++)
287             {
288                 customers[i].remaining_time = customers[i].transaction_time;
289             }
290
291             round_robin(customers, n, time_quantum);
292             break;
293         }
294
295         case 5:
296             free(customers); // Proper memory dealloc
297             printf("Exiting program.\n");
298             exit(0);
299
300         default:
301             printf("Invalid choice.\n");
302         }
303     }
304
305     return 0;
306 }
```

```c
    printf("\n--- ROUND ROBIN SCHEDULING ---\n");

    while (completed < n)
    {
        if (front == rear)
        {
            time++;
            for (int i = 0; i < n; i++)
            {
                if (!visited[i] && customers[i].arrival_time <= time)
                {
                    queue[rear++] = i;
                    visited[i] = 1;
                }
            }
            continue;
        }

        int idx = queue[front++];
        int exec_time =
            customers[idx].remaining_time > time_quantum
                ? time_quantum
                : customers[idx].remaining_time;

        customers[idx].remaining_time -= exec_time;
        time += exec_time;

        for (int i = 0; i < n; i++)
        {
            if (!visited[i] && customers[i].arrival_time <= time)
            {
                queue[rear++] = i;
                visited[i] = 1;
            }
        }

        if (customers[idx].remaining_time > 0)
        {
            queue[rear++] = idx;
        }
        else
        {
            completed++;
            customers[idx].turnaround_time =
                time - customers[idx].arrival_time;
            customers[idx].waiting_time =
                customers[idx].turnaround_time -
                customers[idx].transaction_time;

            total_waiting_time += customers[idx].waiting_time;
            total_turnaround_time += customers[idx].turnaround_time;
        }
    }

    printf("ID\tWaiting\tTurnaround\n");
    for (int i = 0; i < n; i++)
    {
        printf("%d\t%d\t\t%d\n",
               customers[i].customer_id,
               customers[i].waiting_time,
               customers[i].turnaround_time);
    }

    printf("\nAverage Waiting Time: %.2f\n",
           total_waiting_time / n);
    printf("Average Turnaround Time: %.2f\n",
           total_turnaround_time / n);
}
```

```c
int main()
{
    int choice;
    int n;

    // Dynamic memory alloc
    Customer *customers = (Customer *)malloc(MAX_CUSTOMERS * sizeof(Customer));

    if (customers == NULL)
    {
        printf("Memory allocation failed.\n");
        return 1;
    }

    while (1)
    {
        printf("\n====== BANK ATM SCHEDULING SYSTEM ======\n");
        printf("1. Enter Customer Details\n");
        printf("2. FCFS Scheduling\n");
        printf("3. Priority Scheduling (Preemptive)\n");
        printf("4. Round Robin Scheduling\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            printf("Enter number of customers (max %d): ", MAX_CUSTOMERS);
            scanf("%d", &n);

            for (int i = 0; i < n; i++)
            {
                printf("\nCustomer %d\n", i + 1);
                customers[i].customer_id = i + 1;

                printf("Arrival Time: ");
                scanf("%d", &customers[i].arrival_time);

                printf("Transaction Duration: ");
                scanf("%d", &customers[i].transaction_time);

                printf("Priority (0 = VIP, 1 = Regular): ");
                scanf("%d", &customers[i].priority);

                customers[i].remaining_time = customers[i].transaction_time;
            }
            break;

        case 2:
            fcfs(customers, n);
            break;

        case 3:
            for (int i = 0; i < n; i++)
            {
                customers[i].remaining_time = customers[i].transaction_time;
            }
            priority_preemptive(customers, n);
            break;

        case 4:
        {
            int time_quantum;
            printf("Enter Time Quantum: ");
            scanf("%d", &time_quantum);

            for (int i = 0; i < n; i++)
            {
                customers[i].remaining_time = customers[i].transaction_time;
            }

```

```
bhargavdhole@Bhargavs-MacBook-Air ATM_Scheduling_OS % gcc atm_scheduling.c -o atm
bhargavdhole@Bhargavs-MacBook-Air ATM_Scheduling_OS % ./atm

====== BANK ATM SCHEDULING SYSTEM ======
1. Enter Customer Details
2. FCFS Scheduling
3. Priority Scheduling (Preemptive)
4. Round Robin Scheduling
5. Exit
Enter your choice: 1
Enter number of customers (max 50): 3

Customer 1
Arrival Time: 1
Transaction Duration: 1
Priority (0 = VIP, 1 = Regular): 1

Customer 2
Arrival Time: 2
Transaction Duration: 2
Priority (0 = VIP, 1 = Regular): 1

Customer 3
Arrival Time: 33
Transaction Duration: 3
Priority (0 = VIP, 1 = Regular): 0

====== BANK ATM SCHEDULING SYSTEM ======
1. Enter Customer Details
2. FCFS Scheduling
3. Priority Scheduling (Preemptive)
4. Round Robin Scheduling
5. Exit
Enter your choice: 2

--- FCFS SCHEDULING ---
ID      Arrival Burst   Waiting Turnaround
1       1       1       0               1
2       2       2       0               2
3       33      3       0               3

Average Waiting Time: 0.00
Average Turnaround Time: 2.00
```

Thank You.