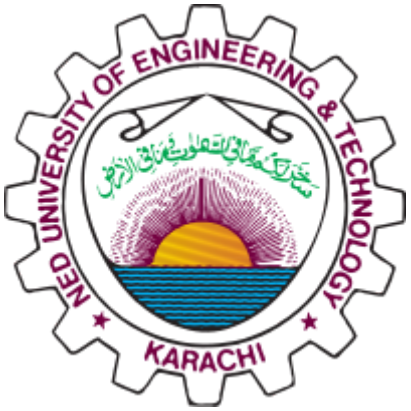
# LAB 02

## Course: CT-353-Operating Systems

## Department: BCIT (Specialisation in Data Science)

## Instructor's Name: Muhammad Muhashir Khan


## Student Name: Maryam Ashraff (DT-22050)

## 3. Round Robin (RR) Scheduling Algorithm

```c
#include <stdio.h>

int main() {
    int i, j, n, bu[10], wa[10], tat[10], t, ct[10], max;
    float awt = 0, att = 0, temp = 0;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        printf("\nEnter Burst Time for process %d: ", i + 1);
        scanf("%d", &bu[i]);
        ct[i] = bu[i];  // Copy burst times for later calculations
    }

    printf("\nEnter time quantum: ");
    scanf("%d", &t);

    max = bu[0];
    for (i = 1; i < n; i++)
        if (max < bu[i])
            max = bu[i];

    for (j = 0; j < (max / t) + 1; j++) {
        for (i = 0; i < n; i++) {
            if (bu[i] != 0) {
                if (bu[i] <= t) {
                    tat[i] = temp + bu[i];
                    temp += bu[i];
                    bu[i] = 0;
                } else {
                    bu[i] -= t;
                    temp += t;
                }
            }
        }
    }

    for (i = 0; i < n; i++) {
        wa[i] = tat[i] - ct[i];
        att += tat[i];
        awt += wa[i];
    }

    printf("\nThe Average Turnaround Time is: %.2f", att / n);
    printf("\nThe Average Waiting Time is: %.2f", awt / n);

    printf("\n\nPROCESS\t BURST TIME\t WAITING TIME\t TURNAROUND TIME\n");
    for (i = 0; i < n; i++)
        printf("%d\t\t %d\t\t %d\t\t %d\n", i + 1, ct[i], wa[i], tat[i]);

    return 0;
}
```

## 4. Priority Scheduling Algorithm

```c
#include <stdio.h>

int main() {
    int p[20], bt[20], pri[20], wt[20], tat[20], i, k, n, temp;
    float wtavg, tatavg;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        p[i] = i;
        printf("Enter the Burst Time & Priority of Process %d: ", i);
        scanf("%d %d", &bt[i], &pri[i]);
    }

    // Sorting based on priority (Lower number = Higher Priority)
    for (i = 0; i < n; i++) {
        for (k = i + 1; k < n; k++) {
            if (pri[i] > pri[k]) {  // Swap if a process has higher priority value
                temp = p[i]; p[i] = p[k]; p[k] = temp;
                temp = bt[i]; bt[i] = bt[k]; bt[k] = temp;
                temp = pri[i]; pri[i] = pri[k]; pri[k] = temp;
            }
        }
    }

    wt[0] = wtavg = 0;
    tat[0] = tatavg = bt[0];

    for (i = 1; i < n; i++) {
        wt[i] = wt[i - 1] + bt[i - 1];
        tat[i] = tat[i - 1] + bt[i];
        wtavg += wt[i];
        tatavg += tat[i];
    }

    printf("\nPROCESS\t PRIORITY\t BURST TIME\t WAITING TIME\t TURNAROUND TIME\n");
    for (i = 0; i < n; i++)
        printf("%d\t\t %d\t\t %d\t\t %d\t\t %d\n", p[i], pri[i], bt[i], wt[i], tat[i]);

    printf("\nAverage Waiting Time: %.2f", wtavg / n);
    printf("\nAverage Turnaround Time: %.2f\n", tatavg / n);

    return 0;
}
```