# Slip 17

## Program 1: LRU Page Replacement Algorithm

```
#include <stdio.h>

#define MAX 20


int frames[MAX], ref[MAX], mem[MAX][MAX], time[MAX], faults = 0, m, n, counter = 0;


void accept() {
   printf("Enter number of frames: ");
   scanf("%d", &n);
   printf("Enter number of references: ");
   scanf("%d", &m);
   printf("Enter reference string:\n");
   for (int i = 0; i < m; i++) {
      printf("[%d] = ", i);
      scanf("%d", &ref[i]);
   }
}


int search(int pno) {
   for (int i = 0; i < n; i++) {
      if (frames[i] == pno) return i;
   }
   return -1;
}


int get_lru() {
   int min = 9999, min_i = 0;
   for (int i = 0; i < n; i++) {
      if (time[i] < min) {
         min = time[i];
         min_i = i;
      }
   }
   return min_i;
}


void lru() {
```

```
    for (int i = 0; i < m; i++) {
        int k = search(ref[i]);
        if (k == -1) {
            if (counter < n) {
                frames[counter] = ref[i];
                time[counter] = i;
                counter++;
            } else {
                int pos = get_lru();
                frames[pos] = ref[i];
                time[pos] = i;
            }
            faults++;
        } else {
            time[k] = i;
        }
        for (int j = 0; j < n; j++) {
            mem[j][i] = frames[j];
        }
    }
}

void disp() {
    printf("\nReference String:\n");
    for (int i = 0; i < m; i++) {
        printf("%3d", ref[i]);
    }
    printf("\n\nFrame Allocation:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (mem[i][j]) {
                printf("%3d", mem[i][j]);
            }
```

# Program 2: FCFS Scheduling Algorithm

```
#include <stdio.h>

struct process {
    int pid;
```

```
    int burst_time;
    int waiting_time;
    int turnaround_time;
};

void calculate_fcfs(struct process p[], int n) {
    int total_waiting = 0, total_turnaround = 0;

    p[0].waiting_time = 0;
    for (int i = 1; i < n; i++) {
        p[i].waiting_time = p[i-1].waiting_time + p[i-1].burst_time;
    }

    for (int i = 0; i < n; i++) {
        p[i].turnaround_time = p[i].waiting_time + p[i].burst_time;
        total_waiting += p[i].waiting_time;
        total_turnaround += p[i].turnaround_time;
    }

    printf("\nPID\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t\t%d\t\t%d\n", p[i].pid, p[i].burst_time, p[i].waiting_time, p[i].turnaround_time);
    }

    printf("\nAverage Waiting Time: %.2f", (float)total_waiting / n);
    printf("\nAverage Turnaround Time: %.2f", (float)total_turnaround / n);
}

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

    struct process p[n];
    for (int i = 0; i < n; i++) {
        p[i].pid = i + 1;
        printf("Enter burst time for process %d: ", p[i].pid);
        scanf("%d", &p[i].burst_time);
    }
```

```
    calculate_fcfs(p, n);


    return 0;
}
```