Exp – 6a):

AIM: To implement First-come First- serve (FCFS) scheduling technique

PROGRAM:

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

int main(){
        int n, avg_wt=0, avg_tat=0;
        printf("enter the number of processes: ");
        scanf("%d", &n);
        int prs[n],bt[n],wt[n],tat[n];
        for(int i=0; i<n; i++){
                printf("process %d\n",i);
                prs[i]=i;
                printf("enter the burst time: ");
                scanf("%d", &bt[i]);
        }
        for(int i=0; i<n; i++){
                if(i==0){
                        wt[i]=0;
                }else{
                        wt[i]=wt[i-1]+bt[i-1];
                }
                tat[i]=bt[i]+wt[i];

                avg_wt += wt[i];
                avg_tat += tat[i];

                printf("process %d\n", i);
                printf("Burst time: %d\n", bt[i]);
                printf("Wait Time: %d\n", wt[i]);
                printf("Turn around time: %d\n", tat[i]);
                printf("\n");
        }
        float awt=avg_wt/n;
        float atat=avg_tat/n;

        printf("average wait time: %.2f\n", awt);
        printf("Average turn around time: %.2f\n", atat);
}
```

OUTPUT:

```
jagadesh@LAPTOP-33VRBQ67:/mnt/c/Users/Parthiban/OS Exps/shell/C programs$ ./fcfs
enter the number of processes: 4
process 0
enter the burst time: 3
process 1
enter the burst time: 5
process 2
enter the burst time: 4
process 3
enter the burst time: 2
process 0
Burst time: 3
Wait Time: 0
Turn around time: 3

process 1
Burst time: 5
Wait Time: 3
Turn around time: 8

process 2
Burst time: 4
Wait Time: 8
Turn around time: 12

process 3
Burst time: 2
Wait Time: 12
Turn around time: 14

average wait time: 5.00
Average turn around time: 9.00
```

Exp – 6b):

AIM: To implement the Shortest Job First (SJF) scheduling technique

PROGRAM:

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

int main(){
    int n, avg_wt = 0, avg_tat = 0;
    printf("Enter the number of processes: ");
    scanf("%d", &n);

    int prs[n], bt[n], wt[n], tat[n];

    for(int i = 0; i < n; i++){
        printf("Process %d\n", i);
        prs[i] = i;
        printf("Enter the burst time: ");
        scanf("%d", &bt[i]);
    }

    for(int i = 0; i < n - 1; i++){
        for(int j = i + 1; j < n; j++){
            if(bt[i] > bt[j]){
                int temp = bt[i]; bt[i] = bt[j]; bt[j] = temp;
                temp = prs[i]; prs[i] = prs[j]; prs[j] = temp;
            }
        }
    }

    for(int i = 0; i < n; i++){
        if(i == 0){
            wt[i] = 0;
        } else {
            wt[i] = wt[i-1] + bt[i-1];
        }
        tat[i] = bt[i] + wt[i];

        avg_wt += wt[i];
        avg_tat += tat[i];

        printf("\nProcess %d\n", prs[i]);
        printf("Burst time: %d\n", bt[i]);
        printf("Wait Time: %d\n", wt[i]);
        printf("Turn around time: %d\n", tat[i]);
        printf("\n");
    }

    float awt = (float)avg_wt / n;
    float atat = (float)avg_tat / n;
    printf("Average wait time: %.2f\n", awt);
    printf("Average turn around time: %.2f\n", atat);
}
```

OUTPUT:

```
jagadesh@LAPTOP-33VRBQ67:/mnt/c/Users/Parthiban/OS Exps/shell/C programs$ ./sfj
Enter the number of processes: 4
Process 0
Enter the burst time: 4
Process 1
Enter the burst time: 6
Process 2
Enter the burst time: 2
Process 3
Enter the burst time: 3

Process 2
Burst time: 2
Wait Time: 0
Turn around time: 2


Process 3
Burst time: 3
Wait Time: 2
Turn around time: 5


Process 0
Burst time: 4
Wait Time: 5
Turn around time: 9


Process 1
Burst time: 6
Wait Time: 9
Turn around time: 15

Average wait time: 4.00
Average turn around time: 7.75
```

Exp – 6c):

AIM: To implement priority scheduling technique

PROGRAM:

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

int main(){
    int n, avg_wt = 0, avg_tat = 0;
    printf("Enter the number of processes: ");
    scanf("%d", &n);

    int prs[n], bt[n], priority[n], wt[n], tat[n];

    for(int i = 0; i < n; i++){
        printf("Process %d\n", i);
        prs[i] = i;
        printf("Enter the burst time: ");
        scanf("%d", &bt[i]);
        printf("Enter the priority: ");
        scanf("%d", &priority[i]);
    }

    for(int i = 0; i < n - 1; i++){
        for(int j = i + 1; j < n; j++){
            if(priority[i] > priority[j]){
                int temp = priority[i]; priority[i] = priority[j]; priority[j] = temp;
                temp = bt[i]; bt[i] = bt[j]; bt[j] = temp;
                temp = prs[i]; prs[i] = prs[j]; prs[j] = temp;
            }
        }
    }

    for(int i = 0; i < n; i++){
        if(i == 0){
            wt[i] = 0;
        } else {
            wt[i] = wt[i-1] + bt[i-1];
        }
        tat[i] = bt[i] + wt[i];

        avg_wt += wt[i];
        avg_tat += tat[i];

        printf("\nProcess %d\n", prs[i]);
        printf("Burst time: %d\n", bt[i]);
        printf("Priority: %d\n", priority[i]);
        printf("Wait Time: %d\n", wt[i]);
        printf("Turn around time: %d\n", tat[i]);
        printf("\n");
    }

    float awt = (float)avg_wt / n;
    float atat = (float)avg_tat / n;

    printf("Average wait time: %.2f\n", awt);
    printf("Average turn around time: %.2f\n", atat);
}
```

OUTPUT:

```
jagadesh@LAPTOP-33VRBQ67:/mnt/c/Users/Parthiban/OS Exps/shell/C programs$ ./priority
Enter the number of processes: 4
Process 0
Enter the burst time: 7
Enter the priority: 1
Process 1
Enter the burst time: 3
Enter the priority: 4
Process 2
Enter the burst time: 5
Enter the priority: 3
Process 3
Enter the burst time: 8
Enter the priority: 2

Process 0
Burst time: 7
Priority: 1
Wait Time: 0
Turn around time: 7


Process 3
Burst time: 8
Priority: 2
Wait Time: 7
Turn around time: 15


Process 2
Burst time: 5
Priority: 3
Wait Time: 15
Turn around time: 20


Process 1
Burst time: 3
Priority: 4
Wait Time: 20
Turn around time: 23

Average wait time: 10.50
Average turn around time: 16.25
```

Exp – 6d):

AIM: To implement the Round Robin (RR) scheduling technique

PROGRAM:

```c
#include <stdio.h>

void roundRobin(int processes[], int n, int at[], int bt[], int quantum) {
    int wt[n], tat[n], rem_bt[n], t = 0, completed = 0;
    float avg_wt = 0, avg_tat = 0;

    for (int i = 0; i < n; i++)
        rem_bt[i] = bt[i];

    while (completed < n) {
        int done = 1;
        for (int i = 0; i < n; i++) {
            if (rem_bt[i] > 0 && at[i] <= t) {
                done = 0;
                if (rem_bt[i] > quantum) {
                    t += quantum;
                    rem_bt[i] -= quantum;
                } else {
                    t += rem_bt[i];
                    wt[i] = t - at[i] - bt[i];
                    rem_bt[i] = 0;
                    completed++;
                }
            }
        }
        if (done) t++;
    }

    for (int i = 0; i < n; i++)
        tat[i] = bt[i] + wt[i];

    printf("\nProcess\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for (int i = 0; i < n; i++) {
        avg_wt += wt[i];
        avg_tat += tat[i];
        printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n", processes[i], at[i], bt[i], wt[i], tat[i]);
    }

    printf("\nAverage Waiting Time: %.2f", avg_wt / n);
    printf("\nAverage Turnaround Time: %.2f\n", avg_tat / n);
}

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

    int processes[n], at[n], bt[n];
    for (int i = 0; i < n; i++) {
        processes[i] = i + 1;
        printf("Enter arrival time for process %d: ", i + 1);
        scanf("%d", &at[i]);
        printf("Enter burst time for process %d: ", i + 1);
        scanf("%d", &bt[i]);
    }

    printf("Enter time quantum: ");
    scanf("%d", &quantum);

    roundRobin(processes, n, at, bt, quantum);
    return 0;
}
```

OUTPUT:

```
jagadesh@LAPTOP-33VRBQ67:/mnt/c/Users/Parthiban/OS Exps/shell/C programs$ ./roundRobin
Enter number of processes: 4
Enter arrival time for process 1: 2
Enter burst time for process 1: 5
Enter arrival time for process 2: 0
Enter burst time for process 2: 3
Enter arrival time for process 3: 4
Enter burst time for process 3: 8
Enter arrival time for process 4: 6
Enter burst time for process 4: 7
Enter time quantum: 4

Process Arrival Time    Burst Time      Waiting Time    Turnaround Time
1               2               5               9               14
2               0               3               0               3
3               4               8               8               16
4               6               7               10              17

Average Waiting Time: 6.75
Average Turnaround Time: 12.50
```