SJF Pre-emptive:

Program:

```
#include<stdio.h>

int main() {
	int N = 5;
	int pid;

	int id[N];
	int at[N];
	int bt[N];

	id[0] = 1;
	id[1] = 2;
	id[2] = 3;
	id[3] = 4;
	id[4] = 5;

	at[0] = 5;
	at[1] = 0;
	at[2] = 13;
	at[3] = 9;
	at[4] = 8;

	bt[0] = 3;
	bt[1] = 9;
	bt[2] = 17;
	bt[3] = 12;
	bt[4] = 2;

	int startingTime[N];
	int endingTime[N] ;
	int btt[N] ;
	for (int i = 0; i < N; i++) {
	startingTime[i] = -1;
	endingTime[i] = -1;
	btt[i] = bt[i];
	}
	double TTAT = 0; // total turnaround time
	double TWT = 0; // total waiting time
	int cycle = 1;

	int totalTime = 0;
```

```
//sorting
for (int j = 0; j < N; j++) {
for (int k = 0; k < N - j - 1; k++) {
if (at[k] > at[k + 1]) {
        int temp = at[k];
        at[k] = at[k + 1];
        at[k + 1] = temp;

        temp = bt[k];
        bt[k] = bt[k + 1];
        bt[k + 1] = temp;

        temp = btt[k];
        btt[k] = btt[k + 1];
        btt[k + 1] = temp;

        temp = id[k];
        id[k] = id[k + 1];
        id[k + 1] = temp;
}
}
}

printf("\nProcesses:\n");
printf("ID\tAT\tBT\n");
printf("------------------\n");
for (int i = 0; i < N; i++) {
printf("%d\t", id[i]);
printf("%d\t", at[i]);
printf("%d\n", bt[i]);
totalTime += bt[i];
}
printf("\n\nTotal time required to run all processes: %d\n", totalTime);
int updatePtr = 0;
int newArrived = 0;
int ce = 0;

while(cycle <= totalTime) {
if(cycle == at[newArrived]) {
printf("\nnew process arrived: pid=%d",id[newArrived]);

newArrived++;

// sorting by burst time
for (int j = 0; j < N; j++) {
        for (int k = 0; k < N - j - 1; k++) {
        if ( (bt[k] > bt[k + 1]) && (at[k + 1] <= cycle) ) {
        int temp = at[k];
```

```
            at[k] = at[k + 1];
            at[k + 1] = temp;

            temp = bt[k];
            bt[k] = bt[k + 1];
            bt[k + 1] = temp;

            temp = btt[k];
            btt[k] = btt[k + 1];
            btt[k + 1] = temp;

            temp = id[k];
            id[k] = id[k + 1];
            id[k + 1] = temp;
            }
            }
}
}

printf("\nce=%d",ce);

printf("\npid=%d \t bt=%d \t cycle=%d \n", id[ce], bt[ce], cycle);

if (startingTime[id[ce] - 1] == -1) {
startingTime[id[ce] - 1] = cycle-1;
}

bt[ce]--;
if(bt[ce] == 0) {
printf("\n----Ended: p%d", id[ce]);
if (endingTime[id[ce] - 1] == -1) {
            endingTime[id[ce] - 1] = cycle;
}
ce ++;
}

// sorting by burst time
for (int j = 0; j < N; j++) {
for (int k = 0; k < N - j - 1; k++) {
            if ( (bt[k] > bt[k + 1]) && (at[k + 1] <= cycle) ) {
            int temp = at[k];
            at[k] = at[k + 1];
            at[k + 1] = temp;

            temp = bt[k];
            bt[k] = bt[k + 1];
            bt[k + 1] = temp;
```

```
                temp = btt[k];
                btt[k] = btt[k + 1];
                btt[k + 1] = temp;

                temp = id[k];
                id[k] = id[k + 1];
                id[k + 1] = temp;
                }
        }
        }
        cycle++;
        }


printf("\n\n\n==========================================================================
==================\n");
        printf("\nProcesses:\n");
        printf("ID\tAT\tBTT\tBT\tST\tET\tTT\tWT\n");
        printf("-------------------------------------------------------\n");
        int tt; // turnaround time
        int wt;
        for (int i = 0; i < N; i++) {
        printf("%d\t", id[i]);
        printf("%d\t", at[i]);
        printf("%d\t", btt[i]);
        printf("%d\t", bt[i]);
        printf("%d\t", startingTime[id[i] - 1]);
        printf("%d\t", endingTime[id[i] - 1]);

        tt = endingTime[id[i] - 1] - at[i];
        TTAT += tt;

        wt = abs(tt - btt[i]);
        TWT += wt;
        printf("%d\t", tt);
        printf("%d\n", wt);
        }

        // getting average
        TTAT = TTAT / N;
        TWT = TWT / N;
        printf("\nTotal Turnaround Time : %f", TTAT);
        printf("\nTotal Waiting Time : %f", TWT);
        printf("\nTotal time required to run all processes: %d\n", totalTime);
        printf("\n=====================================\n");

        return 0;
}
```

Output:

```
i-raj-shinobi-47@irajshinobi47-Lenovo-G50-80:~/Desktop/OSL Practicals/Assignment 3/Shortest Job First$ ./a.out

Processes:
ID    AT    BT
------------------
2    0    9
1    5    3
5    8    2
4    9    12
3    13  17


Total time required to run all processes: 43

==========================================

Processes:
ID    AT    BTT    BT    ST    ET    TT    WT
-----------------------------------------------------------
1    5    3    0    5    8    3    0
5    8    2    0    8    10    2    0
2    0    9    0    0    14    14    5
4    9    12    0    14    26    17    5
3    13    17    0    26    43    30    13

Total Turnaround Time : 13.200000
Total Waiting Time : 4.600000
Total time required to run all processes: 43

==========================================
```