```
#include <stdio.h>
#include <stdlib.h>
struct Process {
    int pid, at, bt, wt, tat, ct, temp, priority;
};

void nppriority (struct Process p[], int number_of_processes) {
    int min, clock = 0;
    int temp = number_of_processes;
    float atat = 0, awt = 0;

    while (temp != 0) {
        min = -1;
        for (int i = 0; i < number_of_processes; i++) {
            if (p[i].at <= clock && p[i].bt != 0) {
                if (min == -1) {
                    min = i;
                    continue;
                }
                if (p[i].priority < p[min].priority) {
                    min = i;
                    continue;
                }

                if (p[i].priority == p[min].priority) {
                    if (p[i].at < p[min].at) {
                        min = i;
                        continue;
                    }

                    if (p[i].at == p[min].at) {
                        if (p[i].pid < p[min].pid)
                            min = i;
                    }
                }
            }
        }

        if (min != -1) {
            clock += p[min].bt;
            p[min].bt = 0;
            p[min].ct = clock;
            p[min].tat = p[min].ct - p[min].at;
            p[min].wt = p[min].tat - p[min].temp;
```

```
            atat += ((float)p[min].tat);
            awt += ((float)p[min].wt);
            temp--;
        } else {
            clock++;
        }
    }

    printf("Following is the table after the execution.\nPID\tPriority\tAT\tBT\tWT\tTAT\tCT\n");
    for (int i = 0; i < number_of_processes; i++) {
        printf("%d\t%d\t\t%d\t%d\t%d\t%d\t%d\n", p[i].pid+1, p[i].priority, p[i].at, p[i].temp, p[i].wt,
p[i].tat, p[i].ct);
    }
    printf("Average waiting time is: %f\n", (awt/((float)number_of_processes)));
    printf("Average turn around time is: %f\n", (atat/((float)number_of_processes)));
}

void ppriority (struct Process p[], int number_of_processes) {
    int min, clock = 0;
    int temp = number_of_processes;
    float atat = 0, awt = 0;

    while (temp != 0) {
        min = -1;
        for (int i = 0; i < number_of_processes; i++) {
            if (p[i].at <= clock && p[i].bt != 0) {
                if (min == -1) {
                    min = i;
                    continue;
                }

                if (p[i].priority < p[min].priority) {
                    min = i;
                    continue;
                }

                if (p[i].priority == p[min].priority) {
                    if (p[i].at < p[min].at) {
                        min = i;
                        continue;
                    }

                    if (p[i].at == p[min].at) {
                        if (p[i].pid < p[min].pid)
```

```
                    min = i;
                }
            }
        }
    }

    if (min != -1) {
        clock++;
        p[min].bt = p[min].bt-1;
        if (p[min].bt == 0) {
            temp--;
            p[min].ct = clock;
            p[min].tat = p[min].ct - p[min].at;
            p[min].wt = p[min].tat - p[min].temp;
            atat += ((float)p[min].tat);
            awt += ((float)p[min].wt);
        }
    } else {
        clock++;
    }
  }

  printf("Following is the table after the execution.\nPID\tPriority\tAT\tBT\tWT\tTAT\tCT\n");
  for (int i = 0; i < number_of_processes; i++) {
    printf("%d\t%d\t\t%d\t%d\t%d\t%d\t%d\n", p[i].pid+1, p[i].priority, p[i].at, p[i].temp, p[i].wt,
p[i].tat, p[i].ct);
  }
  printf("Average waiting time is: %f\n", (awt/((float)number_of_processes)));
  printf("Average turn around time is: %f\n", (atat/((float)number_of_processes)));
}

void round_robin (struct Process p[], int number_of_processes) {
  int time_quantum, clock = 0;
  int temp = number_of_processes;
  float atat = 0, awt = 0;

  printf("Please enter the time quantum: ");
  scanf("%d", &time_quantum);

  int i, counter = 0;
  printf("Following is the table after the execution.\nPID\tAT\tBT\tTAT\tWT\tCT\n");
  for(clock=0, i = 0; temp!=0; ) {
    // define the conditions
    if(p[i].bt <= time_quantum && p[i].bt > 0) {
```

```
            clock = clock + p[i].bt;
            p[i].bt = 0;
            counter = 1;
        }
        else if(p[i].bt > 0) {
            p[i].bt = p[i].bt - time_quantum;
            clock  += time_quantum;
        }
        if(p[i].bt == 0 && counter == 1) {
            temp--; //decrement the process no
            p[i].ct = clock;
            p[i].tat = p[i].ct - p[i].at;
            p[i].wt = p[i].tat - p[i].temp;
            printf("%d\t%d\t%d\t%d\t%d\t%d\n", i+1, p[i].at, p[i].temp, p[i].tat, p[i].wt, p[i].ct);
            awt += ((float)p[i].wt);
            atat += ((float)p[i].tat);
            counter =0;
        }
        if(i == number_of_processes-1)
            i=0;
        else if(p[i+1].at <= clock)
            i++;
        else
            i=0;
    }
    printf("Average waiting time is: %f\n", (awt/((float)number_of_processes)));
    printf("Average turn around time is: %f\n", (atat/((float)number_of_processes)));
}

void main() {
    int number_of_processes, choice;
    printf("Please enter the number of processes you want to enter: ");
    scanf("%d", &number_of_processes);
    struct Process P[number_of_processes];

    while (1) {
        printf("\n\n\n---Menu---\nPlease select the algorithm that you want to use to schedule your
processes.\nNOTE: Lower number means higher priority!\n1. Priority - Non Preemptive\n2.
Priority - Preemptive\n3. Round Robin (RR) - Preemptive\n4. Exit\nPlease enter your choice: ");
        scanf("%d", &choice);

        if (choice == 1) {
            for (int i = 0; i < number_of_processes; i++) {
                printf("Please enter the arrival time for P[%d]: ", i + 1);
```

```
            scanf("%d", &P[i].at);
            printf("Please enter the burst time for P[%d]: ", i + 1);
            scanf("%d", &P[i].bt);
            printf("Please enter the priority for P[%d]: ", i + 1);
            scanf("%d", &P[i].priority);
            P[i].pid = i;
            P[i].temp = P[i].bt;
        }
        nppriority(P, number_of_processes);
        break;
    }

    else if (choice == 2) {
        for (int i = 0; i < number_of_processes; i++) {
            printf("Please enter the arrival time for P[%d]: ", i + 1);
            scanf("%d", &P[i].at);
            printf("Please enter the burst time for P[%d]: ", i + 1);
            scanf("%d", &P[i].bt);
            printf("Please enter the priority for P[%d]: ", i + 1);
            scanf("%d", &P[i].priority);
            P[i].pid = i;
            P[i].temp = P[i].bt;
        }
        ppriority (P, number_of_processes);
        break;
    }

    else if (choice == 3) {
        for (int i = 0; i < number_of_processes; i++) {
            printf("Please enter the arrival time for P[%d]: ", i+1);
            scanf("%d", &P[i].at);
            printf("Please enter the burst time for P[%d]: ", i+1);
            scanf("%d", &P[i].bt);
            P[i].pid = i;
            P[i].temp = P[i].bt;
        }
        round_robin (P, number_of_processes);
        break;
    }

    else if (choice == 4) {
        printf("\nExited!\n");
        exit(1);
    }
```

```
        else {
            printf("\nInvalid Choice!\n");
            break;
        }
    }
}
```