**Operating System Lab-4**

Process creation/ Inter process communication (IPC) Process Creation and Management

Patil Amit Gurusidhappa

19104004

B11

Q1.

#include <stdio.h>

void findWaitingTime(int processes[], int n,

                     int bt[], int wt[])

{

    wt[0] = 0;

    for (int i = 1; i < n; i++)

        wt[i] = bt[i - 1] + wt[i - 1];

}

void findTurnAroundTime(int processes[], int n,

                        int bt[], int wt[], int tat[])

{

    for (int i = 0; i < n; i++)

        tat[i] = bt[i] + wt[i];

}

void findavgTime(int processes[], int n, int bt[])

{

    int wt[n], tat[n], total\_wt = 0, total\_tat = 0;

    findWaitingTime(processes, n, bt, wt);

    findTurnAroundTime(processes, n, bt, wt, tat);

    printf("Processes   Burst time   Waiting time   Turn around time\n");

    for (int i = 0; i < n; i++)

    {

        total\_wt = total\_wt + wt[i];

        total\_tat = total\_tat + tat[i];

        printf("   %d ", (i + 1));

        printf("       %d ", bt[i]);

        printf("       %d", wt[i]);

        printf("       %d\n", tat[i]);

    }

    int s = (float)total\_wt / (float)n;

    int t = (float)total\_tat / (float)n;

    printf("Average waiting time = %d", s);

    printf("\n");

    printf("Average turn around time = %d ", t);

}

int main()

{

    int processes[] = {1, 2, 3};

    int n = sizeof processes / sizeof processes[0];

    int burst\_time[] = {10, 5, 8};

    findavgTime(processes, n, burst\_time);

    return 0;

}

Q2

#include <stdio.h>

void main()

{

    int x, n, p[10], pp[10], pt[10], w[10], t[10], awt, atat, i;

    printf("Enter the number of process : ");

    scanf("%d", &n);

    printf("\n Enter process : time priorities \n");

    for (i = 0; i < n; i++)

    {

        printf("\nProcess no %d : ", i + 1);

        scanf("%d  %d", &pt[i], &pp[i]);

        p[i] = i + 1;

    }

    for (i = 0; i < n - 1; i++)

    {

        for (int j = i + 1; j < n; j++)

        {

            if (pp[i] < pp[j])

            {

                x = pp[i];

                pp[i] = pp[j];

                pp[j] = x;

                x = pt[i];

                pt[i] = pt[j];

                pt[j] = x;

                x = p[i];

                p[i] = p[j];

                p[j] = x;

            }

        }

    }

    w[0] = 0;

    awt = 0;

    t[0] = pt[0];

    atat = t[0];

    for (i = 1; i < n; i++)

    {

        w[i] = t[i - 1];

        awt += w[i];

        t[i] = w[i] + pt[i];

        atat += t[i];

    }

    printf("\n\n Job \t Burst Time \t Wait Time \t Turn Around Time   Priority \n");

    for (i = 0; i < n; i++)

        printf("\n %d \t\t %d  \t\t %d \t\t %d \t\t %d \n", p[i], pt[i], w[i], t[i], pp[i]);

    awt /= n;

    atat /= n;

    printf("\n Average Wait Time : %d \n", awt);

    printf("\n Average Turn Around Time : %d \n", atat);

}

Q3

#include <iostream> using namespace std; int mat[10][6];

void swap(int \*a, int \*b)

{

    int temp = \*a;

    \*a = \*b;

    \*b = temp;

}

void arrangeArrival(int num, int mat[][6])

{

    for (int i = 0; i < num; i++)

    {

        for (int j = 0; j < num - i - 1; j++)

        {

            if (mat[j][1] > mat[j + 1][1])

            {

                for (int k = 0; k < 5; k++)

                {

                    swap(mat[j][k], mat[j + 1][k]);

                }

            }

        }

    }

}

void completionTime(int num, int mat[][6])

{

    int temp, val;

    mat[0][3] = mat[0][1] + mat[0][2];

    mat[0][5] = mat[0][3] - mat[0][1];

    mat[0][4] = mat[0][5] - mat[0][2];

    for (int i = 1; i < num; i++)

    {

        temp = mat[i - 1][3];

        int low = mat[i][2];

        for (int j = i; j < num; j++)

        {

            if (temp >= mat[j][1] && low >= mat[j][2])

            {

                low = mat[j][2];

                val = j;

            }

        }

        mat[val][3] = temp + mat[val][2];

        mat[val][5] = mat[val][3] - mat[val][1];

        mat[val][4] = mat[val][5] - mat[val][2];

        for (int k = 0; k < 6; k++)

        {

            swap(mat[val][k], mat[i][k]);

        }

    }

}

int main()

{

    int num, temp;

    cout << "Enter number of Process: ";

    cin >> num;

    cout << "...Enter the process ID...\n";

    for (int i = 0; i < num; i++)

    {

        cout << "...Process " << i + 1 << "...\n";

        cout << "Enter Process Id: ";

        cin >> mat[i][0];

        cout << "Enter Arrival Time: ";

        cin >> mat[i][1];

        cout << "Enter Burst Time: ";

        cin >> mat[i][2];

    }

    cout << "Before Arrange...\n";

    cout << "Process ID\tArrival Time\tBurst Time\n";

    for (int i = 0; i < num; i++)

    {

        cout << mat[i][0] << "\t\t" << mat[i][1] << "\t\t"

             << mat[i][2] << "\n";

    }

    arrangeArrival(num, mat);

    completionTime(num, mat);

    cout << "Final Result...\n";

    cout << "Process ID\tArrival Time\tBurst Time\tWaiting "

            "Time\tTurnaround Time\n";

    for (int i = 0; i < num; i++)

    {

        cout << mat[i][0] << "\t\t" << mat[i][1] << "\t\t"

             << mat[i][2] << "\t\t" << mat[i][4] << "\t\t"

             << mat[i][5] << "\n";

    }

}

Q4

#include <iostream> using namespace std;

void findWaitingTime(int processes[], int n,

                     int bt[], int wt[], int quantum)

{

    int rem\_bt[n];

    for (int i = 0; i < n; i++)

        rem\_bt[i] = bt[i];

    int t = 0;

    while (1)

    {

        bool done = true;

        for (int i = 0; i < n; i++)

        {

            if (rem\_bt[i] > 0)

            {

                done = false; // There is a pending process

                if (rem\_bt[i] > quantum)

                {

                    t += quantum;

                    rem\_bt[i] -= quantum;

                }

                else

                {

                    t = t + rem\_bt[i];

                    wt[i] = t - bt[i];

                    rem\_bt[i] = 0;

                }

            }

        }

        if (done == true)

            break;

    }

}

void findTurnAroundTime(int processes[], int n,

                        int bt[], int wt[], int tat[])

{

    for (int i = 0; i < n; i++)

        tat[i] = bt[i] + wt[i];

}

void findavgTime(int processes[], int n, int bt[],

                 int quantum)

{

    int wt[n], tat[n], total\_wt = 0, total\_tat = 0;

    findWaitingTime(processes, n, bt, wt, quantum);

    findTurnAroundTime(processes, n, bt, wt, tat);

    cout << "Processes "

         << " Burst time "

         << " Waiting time "

         << " Turn around time\n";

    for (int i = 0; i < n; i++)

    {

        total\_wt = total\_wt + wt[i];

        total\_tat = total\_tat + tat[i];

        cout << " " << i + 1 << "\t\t" << bt[i] << "\t "

             << wt[i] << "\t\t " << tat[i] << endl;

    }

    cout << "Average waiting time = "

         << (float)total\_wt / (float)n;

    cout << "\nAverage turn around time = "

         << (float)total\_tat / (float)n;

}

int main()

{

    int processes[] = {1, 2, 3};

    int n = sizeof processes / sizeof processes[0];

    int burst\_time[] = {10, 5, 8};

    int quantum = 2;

    findavgTime(processes, n, burst\_time, quantum);

    return 0;

}