1. Write a program to compute the average waiting time and average turnaround time based on Non Preemptive Shortest-Job-First Scheduling for the following process with the given CPU burst times, ( and the assumption that all jobs arrive at the same time.)

Process Burst Time

P1 6

P2 8

P3 7

P4 3

Program:

#include <iostream>

using namespace std;

int main() {

int A[100][4];

int i, j, n, total = 0, index, temp;

float avg\_wt, avg\_tat;

cout << "Enter number of process: ";

cin >> n;

cout << "Enter Burst Time:" << endl;

for (i = 0; i < n; i++) {

cout << "P" << i + 1 << ": ";

cin >> A[i][1];

A[i][0] = i + 1;

}

for (i = 0; i < n; i++) {

index = i;

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

if (A[j][1] < A[index][1])

index = j;

temp = A[i][1];

A[i][1] = A[index][1];

A[index][1] = temp;

temp = A[i][0];

A[i][0] = A[index][0];

A[index][0] = temp;

}

A[0][2] = 0;

for (i = 1; i < n; i++) {

A[i][2] = 0;

for (j = 0; j < i; j++)

A[i][2] += A[j][1];

total += A[i][2];

}

avg\_wt = (float)total / n;

total = 0;

cout << "P BT WT TAT" << endl;

for (i = 0; i < n; i++) {

A[i][3] = A[i][1] + A[i][2];

total += A[i][3];

cout << "P" << A[i][0] << " " << A[i][1] << " " << A[i][2] << " " << A[i][3] << endl;

}

avg\_tat = (float)total / n;

cout << "Average Waiting Time= " << avg\_wt << endl;

cout << "Average Turnaround Time= " << avg\_tat << endl;

}

Output:

Enter number of process: 4

Enter Burst Time:

P1: 6

P2: 8

P3: 7

P4: 3

P BT WT TAT

P4 3 0 3

P1 6 3 9

P3 7 9 16

P2 8 16 24

Average Waiting Time= 7

Average Turnaround Time= 13

1. Write a C program to implement the first-fit algorithm for memory management.

Test Case:

Memory partitions: 300 KB, 600 KB, 350 KB, 200 KB, 750 KB, and 125 KB (in order) Show the outcome for the test case with first-fit algorithms to place the processes of size 115 KB, 500 KB, 358 KB, 200 KB, and 375 KB (in order)

Program:

#include<stdio.h>

using namespace std;

void firstFit(int blockSize[], int m,

int processSize[], int n)

{

int allocation[n];

memset(allocation, -1, sizeof(allocation));

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

if (blockSize[j] >= processSize[i])

{

allocation[i] = j;

blockSize[j] -= processSize[i];

break;

}

}

}

cout << "\nProcess No.\tProcess Size\tBlock no.\n";

for (int i = 0; i < n; i++)

{

cout << " " << i+1 << "\t\t"

<< processSize[i] << "\t\t";

if (allocation[i] != -1)

cout << allocation[i] + 1;

else

cout << "Not Allocated";

cout << endl;

}

}

int main()

{

int blockSize[] = {300,600,350,200,750,125};

int processSize[] = {115,500,358,200,375};

int m = sizeof(blockSize) / sizeof(blockSize[0]);

int n = sizeof(processSize) / sizeof(processSize[0]);

firstFit(blockSize, m, processSize, n);

return 0 ;

}

Output:

Process No. Process Size Block no.

1 212 2

2 417 5

3 112 2

4 426 Not Allocated