Experiment No.5
Process Management: Demonstrate the concept of preemptive scheduling algorithms A. Write a program to implement SJF algorithm B. Write a program to implement Priority algorithm
Name Of Student:-Bhagyashri Kaleni Sutar
Roll No:-75
Date of Performance:
Date of Submission:
Marks:
Sign:

Experiment No 5

Aim: To study and implement preemptive scheduling algorithms SJF and Priority Algorithms

Objective:

a. Write a program to demonstrate Shortest Job First (SJF) Algorithm.
b. Write a program to demonstrate Priority Algorithm.

Theory:

A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms.
These algorithms are either non-preemptive or preemptive. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

Shortest Job First (SJF)
This is also known as the shortest job first, or SJF. This is a non-preemptive, preemptive scheduling algorithm. Best approach to minimize waiting time. Easy to implement in Batch systems where required CPU time is known in advance. Impossible to implement in interactive systems where required CPU time is not known. The processor should know in advance how much time the process will take.

Priority Algorithm:

The **Priority Scheduling Algorithm** is a CPU scheduling technique where each process is assigned a **priority** value, and the CPU is allocated to the process with the **highest priority**. If two processes have the same priority, they are scheduled based on arrival order.

Program 1:SJF

```
#include <stdio.h>
int main()
{
	// Matrix for storing Process Id, Burst
	// Time, Average Waiting Time & Average
	// Turn Around Time.
	int A[100][4];
	int i, j, n, total = 0, index, temp;
	float avg_wt, avg_tat;
	printf("Enter number of process: ");
	scanf("%d", &n);
	printf("Enter Burst Time:\n");
	// User Input Burst Time and alloting Process Id.
	for (i = 0; i < n; i++) {
		printf("P%d: ", i + 1);
		scanf("%d", &A[i][1]);
		A[i][0] = i + 1;
	}
	// Sorting process according to their Burst Time.
	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;
    // Calculation of Waiting Times
    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;
    printf("P     BT    WT    TAT\n");
    // Calculation of Turn Around Time and printing the
    // data.
    for (i = 0; i < n; i++) {
        A[i][3] = A[i][1] + A[i][2];
        total += A[i][3];
```

```
                printf("P%d     %d       %d       %d\n", A[i][0],
                        A[i][1], A[i][2], A[i][3]);
        }
        avg_tat = (float)total / n;
        printf("Average Waiting Time= %f", avg_wt);
        printf("\nAverage Turnaround Time= %f", avg_tat);
}
```

Output:-

```
//Enter number of process: 5
Enter Burst Time:
P1: 3
P2: 5
P3: 2
P4: 7
P5: 8
P       BT      WT      TAT
P3      2       0       2
P1      3       2       5
P2      5       5       10
P4      7       10      17
P5      8       17      25
Average Waiting Time= 6.800000
Average Turnaround Time= 11.800000
```

Program 2:Priority Algorithm

```
/*
 * C program to implement priority scheduling
 */


#include <stdio.h>


//Function to swap two variables
void swap(int *a,int *b)
{
   int temp=*a;
   *a=*b;
   *b=temp;
}
int main()
{
   int n;
   printf("Enter Number of Processes: ");
   scanf("%d",&n);


   // b is array for burst time, p for priority and index for process id
   int b[n],p[n],index[n];
   for(int i=0;i<n;i++)
   {
      printf("Enter Burst Time and Priority Value for Process %d: ",i+1);
      scanf("%d %d",&b[i],&p[i]);
      index[i]=i+1;
   }
   for(int i=0;i<n;i++)
   {
      int a=p[i],m=i;


      //Finding out highest priority element and placing it at its desired position

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

      {

        if(p[j] > a)

        {

          a=p[j];

          m=j;

        }

      }


      //Swapping processes

      swap(&p[i], &p[m]);

      swap(&b[i], &b[m]);

      swap(&index[i],&index[m]);

   }


   // T stores the starting time of process

   int t=0;


   //Printing scheduled process

   printf("Order of process Execution is\n");

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

   {

      printf("P%d is executed from %d to %d\n",index[i],t,t+b[i]);

      t+=b[i];
    }
    printf("\n");
    printf("Process Id     Burst Time   Wait Time    TurnAround Time\n");
    int wait_time=0;
    for(int i=0;i<n;i++)
    {
        printf("P%d          %d          %d          %d\n",index[i],b[i],wait_time,wait_time + b[i]);
        wait_time += b[i];
    }
    return 0;
}
```

output:-

```
//Enter Number of Processes: 7
Enter Burst Time and Priority Value for Process 1: 3
4
Enter Burst Time and Priority Value for Process 2: 5
6
Enter Burst Time and Priority Value for Process 3: 7
5
Enter Burst Time and Priority Value for Process 4: 3
2
Enter Burst Time and Priority Value for Process 5: 2


7
Enter Burst Time and Priority Value for Process 6: 5
6
Enter Burst Time and Priority Value for Process 7: 8
```

```
5

Order of process Execution is

P5 is executed from 0 to 2

P2 is executed from 2 to 7

P6 is executed from 7 to 12

P3 is executed from 12 to 19

P7 is executed from 19 to 27

P1 is executed from 27 to 30

P4 is executed from 30 to 33


Process Id     Burst Time   Wait Time    TurnAround Time

P5         2         0         2

P2         5         2         7

P6         5         7         12

P3         7         12         19

P7         8         19         27

P1         3         27         30

P4         3         30         33
```

Conclusion:

1.Applications of Priority Algorithm?

Ans:-Priority algorithms, particularly when implemented with priority queues, have a wide range of applications across various fields. Here are some key areas:

1. Operating Systems and Task Scheduling:

- **Process Management:**

- Operating systems use priority scheduling to determine which process gets the CPU next. Critical system processes or real-time applications are given higher priorities.
-

- **Real-time Systems:**
  - In systems where timely responses are crucial (e.g., medical devices, industrial control), priority algorithms ensure that high-priority tasks are executed without delay.
- **Quality of Service (QoS):**
  - Network routers use priority queuing to prioritize network traffic, ensuring that time-sensitive data (like video or voice) is transmitted with minimal delay.
  -

2. Network and Communication:

- **Network Routing:**
  - Routers may prioritize certain types of network traffic over others, for example, prioritizing voice over IP (VoIP) packets over regular data packets.
  -
- **Packet Processing:**
  - In network devices, priority queues are used to manage incoming and outgoing packets, ensuring that critical packets are processed first.

3. Algorithms and Data Structures:

- **Dijkstra's Algorithm:**
  - Used for finding the shortest path in a graph. Priority queues are essential for efficiently selecting the next node to explore.
  -
- **Prim's Algorithm:**
  - Used for finding the minimum spanning tree of a graph. Priority queues help in selecting the next edge to add to the tree.
  -

- **A* Search Algorithm:**
    - Used in pathfinding and artificial intelligence. Priority queues help in selecting the most promising paths to explore.
    -
- **Huffman Coding:**
    - Used in data compression. Priority queues help to create optimal prefix codes based on symbol frequencies.
    -

4. Simulation and Modeling:

- **Event-driven Simulations:**
    - In simulations where events occur at different times, priority queues are used to manage the order in which events are processed.
    -
- **Discrete Event Simulation:**
    - Used to model systems where the system state changes at discrete points in time. Priority queues are used to maintain the event schedule.
    -

5. Medical and Emergency Services:

- **Triage:**
    - In emergency rooms, patients are prioritized based on the severity of their injuries or illnesses.
    -
- **Emergency Response:**
    - Dispatch systems may prioritize emergency calls based on the urgency of the situation.

2.Applications of Shortest Job First Algorithm?

Ans:-**1. Batch Processing Systems:**

- SJF is most effective in batch processing environments where the execution times of jobs are known in advance.
- This knowledge allows the scheduler to accurately prioritize jobs, leading to minimal average waiting times.
- For example, large data processing tasks, where the processing time is estimated, can be scheduled using SJF.

2. Offline Scheduling:

- SJF can be used in offline scheduling scenarios where all processes and their burst times are known before execution.
- This allows for optimal scheduling decisions to be made.

3. Specific Types of Real-Time Systems (with limitations):

- In some real-time systems where the execution times of tasks are predictable or bounded, SJF-like scheduling can be used.
- However, strict real-time systems often require more predictable scheduling algorithms than SJF, due to the difficulty of guaranteeing deadlines.

4. Data Processing and File Handling:

- When dealing with files or data processing tasks of varying sizes, SJF-like strategies can be used to optimize throughput.
- For example, if a system needs to process a queue of files, it might prioritize processing smaller files first to reduce overall processing time.

5. Variations and Approximations:

- **Shortest Remaining Time First (SRTF):**
    - SRTF is the preemptive version of SJF, where the currently executing process is preempted if a new process arrives with a shorter remaining burst time.
    -
    - SRTF is used in some operating system schedulers, although it still suffers from the issue of needing accurate burst time predictions.