10. Write a program to implement preemptive priority based scheduling algorithm.

A10.

CODE:

```
#include<iostream>
using namespace std;
class Process
{
	int at;
	int bt;
	int res;
	int tar;
	int wt;
	int rem;
	int priority;
	int pid;
	float avgtt;
	float avgwt;
	public:
		void entry(int n)
		{
			pid=n;
			cout<<"\nEnter priority: ";
			cin>>priority;
			cout<<"\nEnter arrival time: ";
			cin>>at;
			cout<<"\nEnter burst time: ";
			cin>>bt;
			rem=bt;
			res=wt=tar=0;
		}

		void preem_prio(Process p[], int size)
		{
			int remain=size;
			int sum_wait=0, sum_turnaround=0;
			int smallest=0;
			for(int time=0; remain!=0; ++time)
			{
				//cout<<"\nIn time loop!";
				if(p[smallest].rem==0)
```

```
                {
                        for(int j=0; j<size; ++j)
                        {
                                if(p[j].rem>0)
                                {
                                        smallest=j;
                                        break;
                                }
                        }
                }

                for(int i=0;i<size;i++)
                {
                        if(p[i].at<=time && p[i].priority<p[smallest].priority &&
p[i].rem>0)
                        {
                                smallest=i;
                        }
                }
                if(p[smallest].rem==p[smallest].bt)
                {
                        p[smallest].res = time-p[smallest].at;
                }

                p[smallest].rem--;

                if(p[smallest].rem==0)
                {
                        remain--;
                        //cout<<"P[%d\t|\t%dt|\t%d\n"<<smallest+1<<time+1-
at[smallest]<<time+1-at[smallest]-bt[smallest];
                        p[smallest].tar = time+1-p[smallest].at;
                        p[smallest].wt = time+1-p[smallest].at-p[smallest].bt;
                        sum_wait+=time+1-p[smallest].at;
                        sum_turnaround+=time+1-p[smallest].at-p[smallest].bt;
                }
        }
        avgwt = sum_wait/size;
        avgtt = sum_turnaround/size;
}

void show_data(Process pr[], int x)
```

```
		{
			cout<<"\n--------------------------------------------------------------------------------";
			cout<<"\nPid	Arr	Priority	BurstTime	WaitingTime	TurnaroundTime	ResponseTime\n";
			cout<<"\n--------------------------------------------------------------------------------\n";
			for(int c=0; c<x; ++c)

	cout<<pr[c].pid<<"\t"<<pr[c].at<<"\t"<<pr[c].priority<<"\t"<<pr[c].bt<<"\t\t"<<pr[c].wt<<"\t\t"<<pr[c].tar<<"\t\t"<<pr[c].res<<endl;
			cout<<"\nAverage Waiting Time: "<<avgwt;
			cout<<"\nAverage Turnaround Time: "<<avgtt;
		}
};

int main()
{
	Process pro[10];
	int n;
	cout<<"\nEnter no. of processes (max 10) : ";
	cin>>n;
	do
	{
		if(n<0||n>10)
			{
				cout<<"\nEnter again : ";
				cin>>n;
			}
	}while(n<0||n>10);

	for(int i=0; i<n; ++i)
	{
		cout<<"\nEnter details for "<<i<<" process: \n";
		pro[i].entry(i);
	}
	pro[0].preem_prio(pro, n);
	pro[0].show_data(pro, n);

	return 0;
}
```

OUTPUT:

```
Enter no. of processes (max 10) : 4
Enter details for 0 process:
Enter priority: 3
Enter arrival time: 0
Enter burst time: 4
Enter details for 1 process:
Enter priority: 2
Enter arrival time: 1
Enter burst time: 2
Enter details for 2 process:
Enter priority: 4
Enter arrival time: 2
Enter burst time: 3
Enter details for 3 process:
Enter priority: 1
Enter arrival time: 4
Enter burst time: 2
```

Pid	Arr	Priority	BurstTime	WaitingTime	TurnaroundTime	ResponseTime
0	0	3	4	4	8	0
1	1	2	2	0	2	0
2	2	4	3	6	9	6
3	4	1	2	0	2	0

```
Average Waiting Time: 5
Average Turnaround Time: 2
```