AMAN SINGH NEGI
BCA 5B
UNIVERSITY ROLL NO :- 1921018
SUBJECT :- MID SEM BACK OPERATING SYSTEM.

Ques 1. #Include <stdio. h>

```
unsigned int Heap[100001], Index[100001],
Position[100002], Size=0;
unsigned int Temp[100001], Temp1[100001];
unsigned int Arr_Time[100001], Cook_Time
[100001], Num;
void merge(int Low, int Mid, int High)
{
  int i=Low, j=Mid+1, k=0;
 while(i<=Mid && j<=High)
  {
  if (Arr_Time[i]<=Arr_Time[j])
  {
  Temp[k]=Arr_Time[i];
  Temp1[k]=Cook_Time[i];
  i++;
  k++;
  }
```

```
} if (i<=Mid)
  {
  int I;
  for(I=i ; I.<=Mid ; I++)
  {Temp[k] = Arr_Time[I];
  Temp2[k] = Cook_Time[I]; k++; }
}
else if (j<= High)
  {
  int I;
  for(I=j; I<=High; I++)
{Temp[K] = Arr_Time[I]; Temp2[k]=
  Cook time[I]; K++; }
}
  K=0
  for(i=Low; i<=High; i++)
{
  Arr_Time[i] = Temp[k];
  Cook_Time[i] = Temp2[k];
  }
}
```

```
void divide(int low, int High)
{
    if (low < High)
    {
        int Mid = (low + High)/2;
        divide(low, Mid);
        divide(Mid+1, High);
        merge(low, Mid, High);
    }
}
void Insert(int Node, unsigned
                        int value)
{
    int S;
    if (Position[Node] == 0)
    {
        Heap[++Size] = Value;
        Index[Size] = Node;
        Position[Node] = Size;
        S = Size;
    }
    else
```

```
{
Heap[Position[Node]] = Value;
S = Position[Node];
}
while (S != 1)
{
if (Heap[S/2] > Heap[S])
{
int t = Heap[S/2];
Heap[S/2] = Heap[S];
Heap[S] = t;
t = Index[S/2];
Index[S/2] = Index[S];
Index[S] = t;
Position[Index[S/2]] = S/2;
Position[Index[S]] = S;
}
else
break;
S = S/2;
}
}
```

```
int Extract_Min()
{
    int N=Index[1];
    int S=1;
  Position[N]=-1;
  Index[1]=Index[Size];
  Position[Index[Size]]=1;
  Heap[1]=Heap[Size--];
   while(1)
    {
      int T
      if (Heap[S*2]<Heap[S]
  && S*2<=Size || Heap[S*2+1]
  <Heap[S] && S*2+1<=Size)
      {
        if (Heap[S*2]<Heap[S*2+
                           1])
  T= S*2;
  else
   T= S*2+1;
        int t=Heap[T];
  Heap[T]=Heap[S];
```

```
Heap[S]=t;
t=Index[T];
Index[T]=Index[S];
Index[S]=t;

Position[Index[T]]=T;
Position[Index[S]]=S;
}
else
break;
S=T;
}
return N;
}
void Init(int N)
{int i;
for(i=1;i<=N;i++)
{
Position[i]=0;
Index[i]=0
Heap[i]=10000000001;
}
```

```
size=N
}
int main()
{  int A_T, C_T, i=1;
long long wait_Time=0, Time=0;
   scanf("%d", &Num);
for(i=0;i<Num;i++)
scanf("%u%u,&Arr_Time[i],&
    Cook_Time[i]);
divide(0,Num-1);
 for(I=Num;i>=1;i--)
 {
        Arr_Time[i]=Arr_Time[i-1];
Cook Time[i]=Cook_Time[i-1];
 }
Insert(1,Cook_Time[1]);
i=2;
 while(i<=Num && Arr_Time
  [i]==Arr_Time[1]);
```

```
i++;
}
while (size!=0)
{
int I=Extract_Min();
if (Time > Arr_Time[I])
{
wait_Time[I]=Time-Arr_Time[I]
+Cook_Time[I];
Time+Cook_Time[I];
}
else
{
Time=Arr_Time[I]+Cook_
Time[I];
}
I=i;
while (i<=Num && Arr_Time[i]
<=Time)
{
insert(i,Cook_time[i]);
i++;
}if(I==i && i<=Num)
{
```

```
        Insert(i, Cook_Time(i));
        i++;
    While(i<=Num && Arr_Time[i]
    = Arr_Time[I])
    {
    insert(i, Cook_Time(i));
        i++;
    }
    }
}
Wait_Time = Wait_Time/Num;
Printf("%.11d" Wait_Time);
return 0;
}
```

Ques 2.

```
#include

void main()
{
int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
float avg_wt,avg_tat;
printf("Enter number of process:");
scanf("%d",&n);

printf("\nEnter Burst Time:\n");
for(i=0;i<n;i++)
{
printf("p%d:",i+1);
scanf("%d",&bt[i]);
p[i]=i+1;   contains process number
   }

  sorting burst time in ascending order using selection sort
for(i=0;i<n;i++)
{
pos=i;
for(j=i+1;j<n;j++)
{
if(bt[j]<bt[pos])pos=j;
   }

temp=bt[i];
bt[i]=bt[pos];
bt[pos]=temp;

temp=p[i];
```

```
p[i]=p[pos];
p[pos]=temp;
  }
wt[0]=0;  waiting time for first process will be zero
  calculate waiting time
for(i=1;i<n;i++)
wt[i]=0;
for(j=0;j<i;j++) wt[i]+=bt[j];

total+=wt[i];
  }

avg_wt=(float)total/n;  average waiting time
total=0;

printf("\nProcess\tBurst Time \tWaiting Time\tTurnaround
Time");
for(i=0;i<n;i++)
tat[i]=bt[i]+wt[i];  calculate turnaround time
total+=tat[i];
printf("\np%d\t\t %d\t\t %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
  }

avg_tat=(float)total/n;  average turnaround time
printf("\n\nAverage Waiting Time=%f",avg_wt);
printf("\nAverage Turnaround Time=%f\n",avg_tat);
}
```

```
Enter number of process:4

Enter Burst Time:
p1:10
p2:2
p3:1
p4:4

Process      Burst Time          Waiting Time     Turnaround Time
p3                 1                   0                   1
p2                 2                   1                   3
p4                 4                   3                   7
p1                 10                  7                   17

Average Waiting Time=2.750000
Average Turnaround Time=7.000000


...Program finished with exit code 0
Press ENTER to exit console.
```