PROGRAM – 12

```
#include<stdio.h>
struct memory
{
  int size;
  int k;
  int IF;
  int EXF;
  int Fsize;
  int id;
  int i;
  int ps;
}m[10];
struct process
{
    int id;
    int size;
    int k;
}p[10];

int main()
{
    int b,i,pr,j,TIF=0,TEXF=0,max,k,l=0,count=0,psize;
    struct memory temp;
    printf("Enter no of blocks \n");
    scanf("%d",&b);
    printf("Enter the block sizes \n--> ");
    for(i=0;i<b;i++)
    {
        scanf("%d",&m[i].size);
        m[i].k=0;
        m[i].i=i+1;
    }
    printf("\nEnter no of processes \n");
    scanf("%d",&pr);
    printf("Enter the process sizes \n ");
    for(i=0;i<pr;i++)
    {
        printf("P%d ",i+1);
        scanf("%d",&p[i].size);
        p[i].id=i+1;
    }
    for(i=0;i<b;i++)
    {
        k=i;
        max=m[i].size;
        for(j=i;j<b;j++)
        {
            if(max<m[j].size)
            {
                max=m[j].size;
                k=j;
                l++;
            }
        }
        if(l>0)
        {
            temp=m[i];
            m[i]=m[k];
            m[k]=temp;
        }
    }
    for(i=0;i<pr;i++)
    {
        for(j=0;j<b;j++)
        {
            if(m[j].size>=p[i].size && m[j].k==0 && p[i].k==0)
            {
                m[j].id=p[i].id;
                m[j].IF=m[j].size-p[i].size;
                m[j].EXF=0;
                m[j].Fsize=m[j].IF;
                m[j].ps=p[i].size;
                m[j].k=1;
                p[i].k=1;
            }
        }
    }
    for(i=0;i<b;i++)
    {
        if(m[i].k==0)
        {
            m[i].IF=0;
            m[i].id=-1;
        }
    }
    for(i=0;i<pr;i++)
    {
        if(p[i].k==1)
        {
            count=count+1;
        }
        else
        {
            psize=p[i].size;
        }
    }
    for(i=0;i<b;i++)
    {
        for(j=0;j<b;j++)
        {
            if(i+1==m[j].i)
            {
                temp=m[i];
                m[i]=m[j];
                m[j]=temp;
            }
        }
    }
    for(i=0;i<b;i++)
    {
        if(m[i].k==0 && m[i+1].k==0)
        {
            TEXF=TEXF+m[i].size;
        }
        if(i>0)
        {
            if(m[i].k==0 && m[i+1].k==1 && m[i-1].k==0)
            {
                TEXF=TEXF+m[i].size;
            }
        }

    }
    printf("\nBlocks :  ");
    for(i=0;i<b;i++)
    {
        printf("| %d  ",m[i].size);
    }
    printf("\n\nProcesses :  ");
    for(i=0;i<pr;i++)
    {
        printf("|  p[%d]-%d  ",p[i].id,p[i].size);
    }
    printf("\n\n\nBlock No\tSize of Block\tprocess 
allocated\tIF\n\n");
    for(i=0;i<b;i++)
    {
        if(m[i].id!=-1)
        {
            printf("%d\t\t%d\t\tp%d[%d]
\t\t\t%d\n",m[i].i,m[i].size,m[i].id,m[i].ps,m[i].IF);
        }
        else
        {

printf("%d\t\t%d\t\tNULL\t\t\t%d\n",m[i].i,m[i].size,m[i].IF);
        }

    }
    for(i=0;i<b;i++)
    {
        TIF=TIF+m[i].IF;
        TEXF=TEXF+m[i].EXF;
    }
    if(psize > TEXF)
    {
        TEXF=0;
    }
    if(count==pr)
    {
        TEXF=0;
    }
    printf("\nTotal internal fragmentation =  %d\n",TIF);
    printf("\nTotal external fragmentation =  %d\n",TEXF);
}
```