```
/*                  EXPERIMENT NUMBER -> 15
Gaurav Rai(1706019)
Question:
Consider 6 mmemory partition of sizes 200KB,400KB,600KB,500kB,300KB,250KB.This
partition needs to be
allocated to four processess of sizes using 375KB, 210KB, 468KB and 491KB using
1.First fit
2.Best fit
3.Next fit
4.worst fit
*/
#include<bits/stdc++.h>
using namespace std;
struct proces{
    int pno,psize,part_no_alloc;
    bool allocated;
    int blockszal;
};
struct MemPartition
{
    int blockno,blocksz;bool allocated;
};
void PrintPr(vector<proces> Pr,int n)
{
    cout<<"Process Number\tProcess Size\tBlock Allocated\tBlock Size\n";
    for(int i=0;i<n;i++)
    {
        cout<<Pr[i].pno<<"\t\t"<<Pr[i].psize<<"\t\t";
        if(Pr[i].allocated)
        cout<<Pr[i].part_no_alloc<<"\t\t"<<Pr[i].blockszal<<endl;
        else
        cout<<"NA\t\tNA\n";
    }

}

void firstFit(vector<MemPartition> P,int m,vector<proces> Pr,int n)
{
    for(int i=0;i<n;i++){
        for(int j=0;j<m;j++){
            if(!P[j].allocated && P[j].blocksz>=Pr[i].psize){
                P[j].allocated=true;Pr[i].part_no_alloc=P[j].blockno;
                Pr[i].blockszal=P[j].blocksz;
                Pr[i].allocated=true;break;
            }
        }
    }
    PrintPr(Pr,n);
}
void bestFit(vector<MemPartition> P,int m,vector<proces> Pr,int n)
```

```
{
    int tempIndex=-1;
    for(int i=0;i<n;i++){
        tempIndex=-1;
        for(int j=0;j<m;j++){
            if(!P[j].allocated && Pr[i].psize<=P[j].blocksz){
                if(tempIndex==-1)
                tempIndex=j;
                else{
                    if(P[j].blocksz<P[tempIndex].blocksz)
                        tempIndex=j;
                }
            }
        }
        if(tempIndex!=-1)
        {
            P[tempIndex].allocated=true;Pr[i].allocated=true;
            Pr[i].part_no_alloc=P[tempIndex].blockno;
            Pr[i].blockszal=P[tempIndex].blocksz;
        }
    }
    PrintPr(Pr,n);
}
void nextFit(vector<MemPartition> P,int m,vector<proces> Pr,int n)
{
    int tempIndex=0;
    for(int i=0;i<n;i++){
        int ct=0;
        for(int j=tempIndex;ct<m;ct++){
            if(!P[j].allocated && P[j].blocksz>=Pr[i].psize){
                P[j].allocated=true;Pr[i].part_no_alloc=P[j].blockno;
                Pr[i].blockszal=P[j].blocksz;
                Pr[i].allocated=true;
                if(j+1<m)
                tempIndex=j+1;
                else
                tempIndex=0;
                break;
            }
            if(j+1<m)
            j++;
            else
            j=0;
        }
    }
    PrintPr(Pr,n);
}
void worstFit(vector<MemPartition> P,int m,vector<proces> Pr,int n)
{
    int tempIndex=-1;
    for(int i=0;i<n;i++){
        tempIndex=-1;
        for(int j=0;j<m;j++){
            if(!P[j].allocated && Pr[i].psize<=P[j].blocksz){
                if(tempIndex==-1)
                tempIndex=j;
                else{
                    if(P[j].blocksz>P[tempIndex].blocksz)
                        tempIndex=j;
                }
            }
        }
        if(tempIndex!=-1)
        {
            P[tempIndex].allocated=true;Pr[i].allocated=true;
            Pr[i].part_no_alloc=P[tempIndex].blockno;
            Pr[i].blockszal=P[tempIndex].blocksz;
        }
    }
    PrintPr(Pr,n);
}
int main()
{
    int no_of_mem_part;
    cout<<"Enter the number of memory partitions : ";cin>>no_of_mem_part;
    vector<MemPartition> parts(no_of_mem_part);
    cout<<"Enter the size of  : "<<endl;
    for(int i=0;i<no_of_mem_part;i++){
        cout<<"Partition "<<(i)<<" : ";cin>>parts[i].blocksz;
        parts[i].blockno=i;
        parts[i].allocated=false;
    }
    int no_of_pro;
    cout<<"Enter the number of processes ";cin>>no_of_pro;
    vector<proces> process(no_of_pro);
    cout<<"Enter the size of :  "<<endl;
    for(int i=0;i<no_of_pro;i++)
    {
        cout<<"Process "<<i<<" : ";cin>>process[i].psize;
        process[i].pno=i;
        process[i].allocated=false;
    }
    cout<<"\t\tFIRST FIT"<<endl;
    firstFit(parts,no_of_mem_part,process,no_of_pro);
    cout<<endl;
    cout<<"\t\tBEST FIT"<<endl;
    bestFit(parts,no_of_mem_part,process,no_of_pro);
    cout<<endl;
    cout<<"\t\tNEXT FIT"<<endl;
    nextFit(parts,no_of_mem_part,process,no_of_pro);
    cout<<endl;
    cout<<"\t\tWORST FIT"<<endl;
    worstFit(parts,no_of_mem_part,process,no_of_pro);
    return 0;
}
```