19BCE0829
KANDRA KSHEERAJ

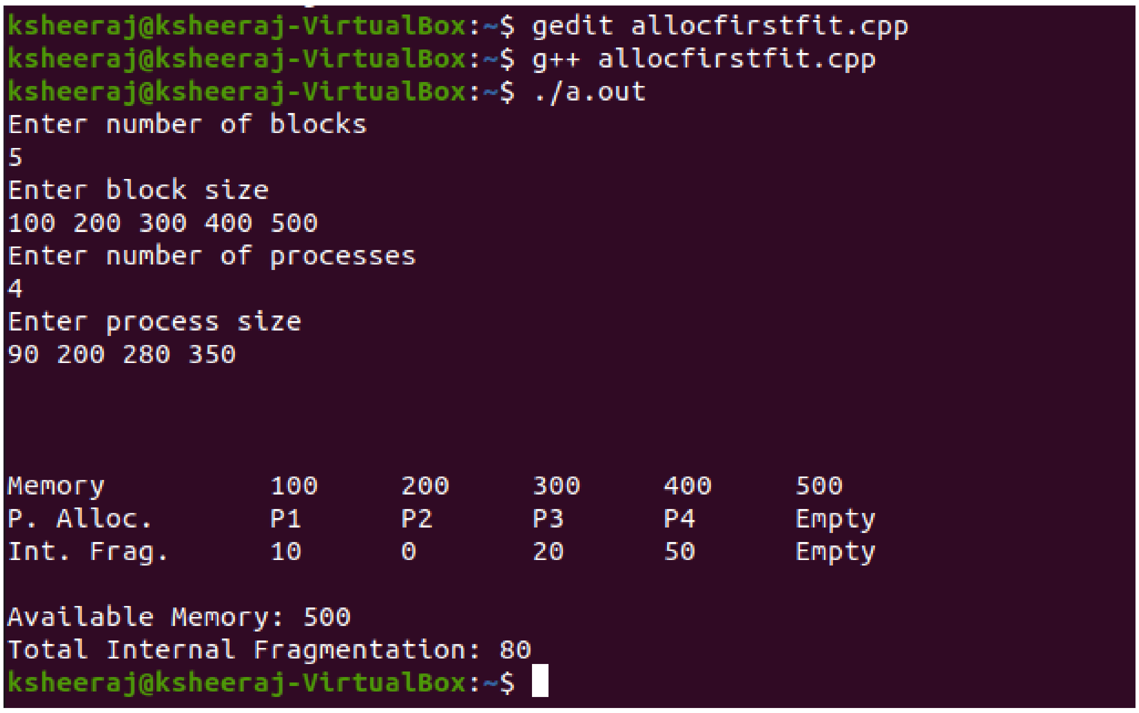
Continuous memory allocation

a) First-Fit

```
#include<iostream>
#include<algorithm>
using namespace std;

struct node{
    int memsize;
    int allocp=-1;
    int pos;

   int allocSize;
}m[200];


bool posSort(node a,node b){
    return a.pos < b.pos;
}

bool memSort(node a,node b){
    return a.memsize < b.memsize;
}

int main(){
    int nm,np,choice, i, j, p[200];
    cout<<"Enter number of blocks\n";
    cin>>nm;
    cout<<"Enter block size\n";
    for(i=0;i<nm;i++){
        cin>>m[i].memsize;
        m[i].pos=i;
    }
    cout<<"Enter number of processes\n";
    cin>>np;
    cout<<"Enter process size\n";
    for(i=0;i<np;i++){
        cin>>p[i];
    }
    cout<<"\n\n";
```

```
//sort(m,m+nm,memSort);
int globalFlag=0;

for(i=0;i<np;i++){
    int flag=0;
    for(j=0;j<nm;j++){
        if(p[i]<=m[j].memsize && m[j].allocp==-1){
            m[j].allocp=i;
            m[j].allocSize=p[i];
            flag=1;
            break;
        }
    }
    if(flag==0){
            cout<<"Unallocated Process P"<<i+1<<"\n";
            globalFlag=1;
        }
    }
sort(m,m+nm,posSort);
cout<<"\n";
int intFrag=0,extFrag=0;
cout<<"Memory\t\t";
for(i=0;i<nm;i++){
    cout<<m[i].memsize<<"\t";
}
cout<<"\n";
cout<<"P. Alloc.\t";
for(i=0;i<nm;i++){
    if(m[i].allocp!=-1){
        cout<<"P"<<m[i].allocp+1<<"\t";
    }
    else{
        cout<<"Empty\t";
    }
}
cout<<"\n";
cout<<"Int. Frag.\t";
for(i=0;i<nm;i++){
        if(m[i].allocp!=-1){
            cout<<m[i].memsize-m[i].allocSize<<"\t";
```

```
                    intFrag+=m[i].memsize-m[i].allocSize;
                }
                else{
                    extFrag+=m[i].memsize;
                    cout<<"Empty\t";
                }
        }
        cout<<"\n";
        cout<<"\n";
        if(globalFlag==1)
            cout<<"Total External Fragmentation: 
    "<<extFrag<<"\n";
        else
        {
            cout<<"Available Memory: "<<extFrag<<"\n";
        }
            cout<<"Total Internal Fragmentation: 
    "<<intFrag<<"\n";
        return 0;
    }
```

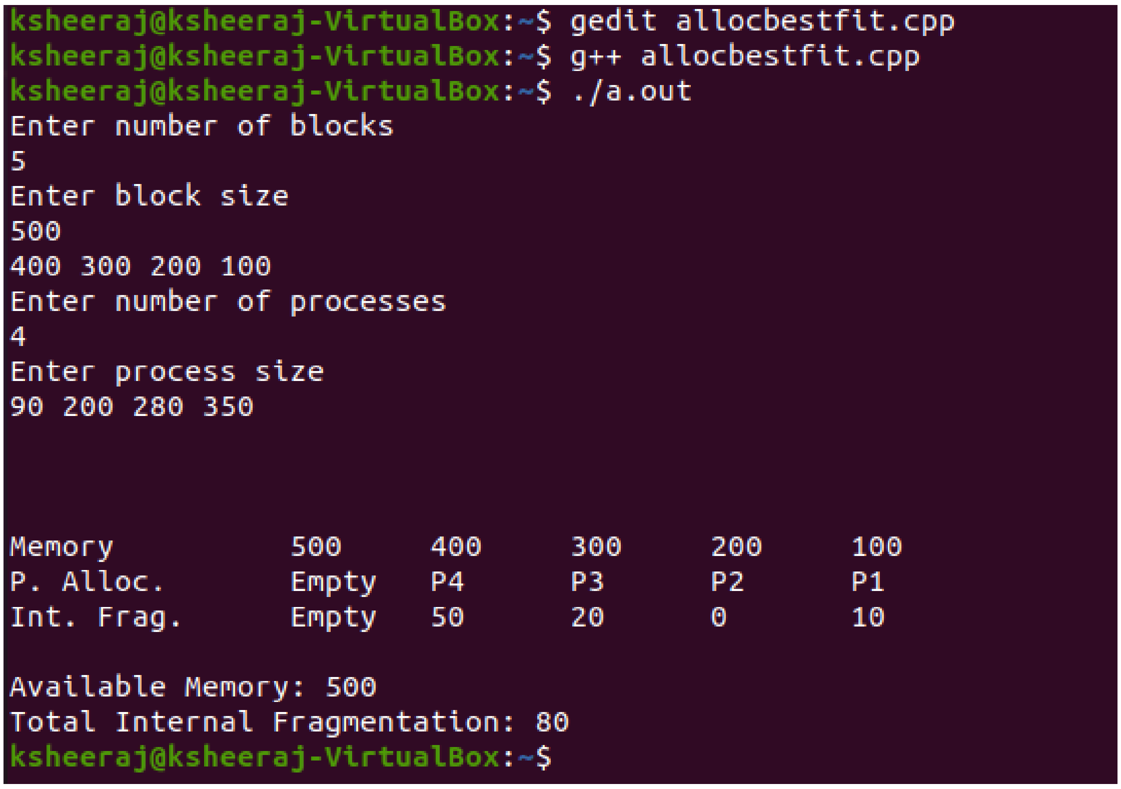


```
ksheeraj@ksheeraj-VirtualBox:~$ gedit allocfirstfit.cpp
ksheeraj@ksheeraj-VirtualBox:~$ g++ allocfirstfit.cpp
ksheeraj@ksheeraj-VirtualBox:~$ ./a.out
Enter number of blocks
5
Enter block size
100 200 300 400 500
Enter number of processes
4
Enter process size
90 200 280 350



Memory          100     200     300     400     500
P. Alloc.       P1      P2      P3      P4      Empty
Int. Frag.      10      0       20      50      Empty

Available Memory: 500
Total Internal Fragmentation: 80
ksheeraj@ksheeraj-VirtualBox:~$ 
```

b) Best-Fit

```
#include<iostream>
#include<algorithm>
```

```
using namespace std;

struct node{
    int memsize;
    int allocp=-1;
    int pos;
    int allocSize;
}m[200];
bool posSort(node a,node b){
    return a.pos < b.pos;
}
bool memSort(node a,node b){
    return a.memsize < b.memsize;
}
int main(){
    int nm,np,choice, i, j, p[200];
    cout<<"Enter number of blocks\n";
    cin>>nm;
    cout<<"Enter block size\n";
    for(i=0;i<nm;i++){
        cin>>m[i].memsize;
        m[i].pos=i;
    }

    cout<<"Enter number of processes\n";
    cin>>np;

    cout<<"Enter process size\n";
    for(i=0;i<np;i++){
        cin>>p[i];
    }
    cout<<"\n\n";
    sort(m,m+nm,memSort);
    int globalFlag=0;

    for(i=0;i<np;i++){
        int flag=0;
        for(j=0;j<nm;j++){
            if(p[i]<=m[j].memsize && m[j].allocp==-1){
                m[j].allocp=i;
                m[j].allocSize=p[i];
                flag=1;
                break;
            }
        }
        if(flag==0){
                cout<<"Unallocated Process P"<<i+1<<"\n";
                globalFlag=1;
            }
        }
    sort(m,m+nm,posSort);
    cout<<"\n";
    int intFrag=0,extFrag=0;
    cout<<"Memory\t\t";
    for(i=0;i<nm;i++){
        cout<<m[i].memsize<<"\t";
    }
    cout<<"\n";
    cout<<"P. Alloc.\t";
    for(i=0;i<nm;i++){
        if(m[i].allocp!=-1){
            cout<<"P"<<m[i].allocp+1<<"\t";
        }
        else{
            cout<<"Empty\t";
        }
    }
    cout<<"\n";
    cout<<"Int. Frag.\t";
    for(i=0;i<nm;i++){
            if(m[i].allocp!=-1){
                cout<<m[i].memsize-m[i].allocSize<<"\t";
                intFrag+=m[i].memsize-m[i].allocSize;
            }
            else{
                extFrag+=m[i].memsize;
                cout<<"Empty\t";
            }
    }
    cout<<"\n";
    cout<<"\n";

    if(globalFlag==1)
        cout<<"Total External Fragmentation: 
"<<extFrag<<"\n";
    else
    {
        cout<<"Available Memory: "<<extFrag<<"\n";
    }

    cout<<"Total Internal Fragmentation: "<<intFrag<<"\n";
    return 0;
}
```



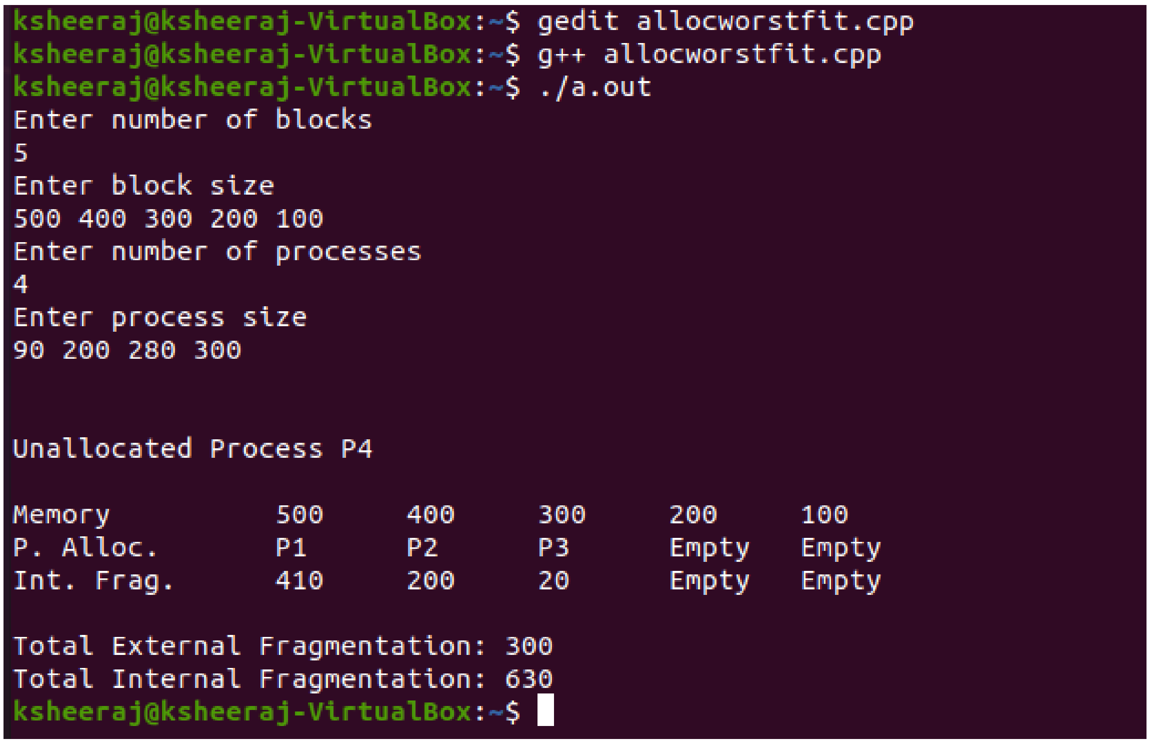
```
ksheeraj@ksheeraj-VirtualBox:~$ gedit allocbestfit.cpp
ksheeraj@ksheeraj-VirtualBox:~$ g++ allocbestfit.cpp
ksheeraj@ksheeraj-VirtualBox:~$ ./a.out
Enter number of blocks
5
Enter block size
500
400 300 200 100
Enter number of processes
4
Enter process size
90 200 280 350



Memory          500     400     300     200     100
P. Alloc.       Empty   P4      P3      P2      P1
Int. Frag.      Empty   50      20      0       10

Available Memory: 500
Total Internal Fragmentation: 80
ksheeraj@ksheeraj-VirtualBox:~$
```

c) Worst-Fit

```
#include<iostream>
#include<algorithm>
using namespace std;

struct node{
    int memsize;
    int allocp=-1;
    int pos;
    int allocSize;
}m[200];


bool posSort(node a,node b){
    return a.pos < b.pos;
}

bool memSort(node a,node b){
    return a.memsize > b.memsize;
}

int main(){
    int nm,np,choice, i, j, p[200];
    cout<<"Enter number of blocks\n";
    cin>>nm;
    cout<<"Enter block size\n";
    for(i=0;i<nm;i++){
        cin>>m[i].memsize;
        m[i].pos=i;
    }
    cout<<"Enter number of processes\n";
    cin>>np;
    cout<<"Enter process size\n";
    for(i=0;i<np;i++){
        cin>>p[i];
    }
    cout<<"\n\n";
    sort(m,m+nm,memSort);
    int globalFlag=0;
    for(i=0;i<np;i++){
        int flag=0;
        for(j=0;j<nm;j++){
            if(p[i]<=m[j].memsize && m[j].allocp==-1){
                m[j].allocp=i;
                m[j].allocSize=p[i];
                flag=1;
                break;
            }
        }
```

```
          if(flag==0){
                  cout<<"Unallocated Process P"<<i+1<<"\n";
                  globalFlag=1;
              }
          }
      sort(m,m+nm,posSort);
      cout<<"\n";
      int intFrag=0,extFrag=0;
      cout<<"Memory\t\t";
      for(i=0;i<nm;i++){
          cout<<m[i].memsize<<"\t";
      }
      cout<<"\n";
      cout<<"P. Alloc.\t";
      for(i=0;i<nm;i++){
          if(m[i].allocp!=-1){
              cout<<"P"<<m[i].allocp+1<<"\t";
          }
          else{
              cout<<"Empty\t";
          }
      }
      cout<<"\n";
      cout<<"Int. Frag.\t";
      for(i=0;i<nm;i++){
              if(m[i].allocp!=-1){
                  cout<<m[i].memsize-m[i].allocSize<<"\t";
                  intFrag+=m[i].memsize-m[i].allocSize;
              }
              else{
                  extFrag+=m[i].memsize;
                  cout<<"Empty\t";
              }
      }
      cout<<"\n";
      cout<<"\n";

      if(globalFlag==1)
          cout<<"Total External Fragmentation:
"<<extFrag<<"\n";
```

```
    else{
        cout<<"Available Memory: "<<extFrag<<"\n";
    }
    cout<<"Total Internal Fragmentation: "<<intFrag<<"\n";
    return 0;
}
```

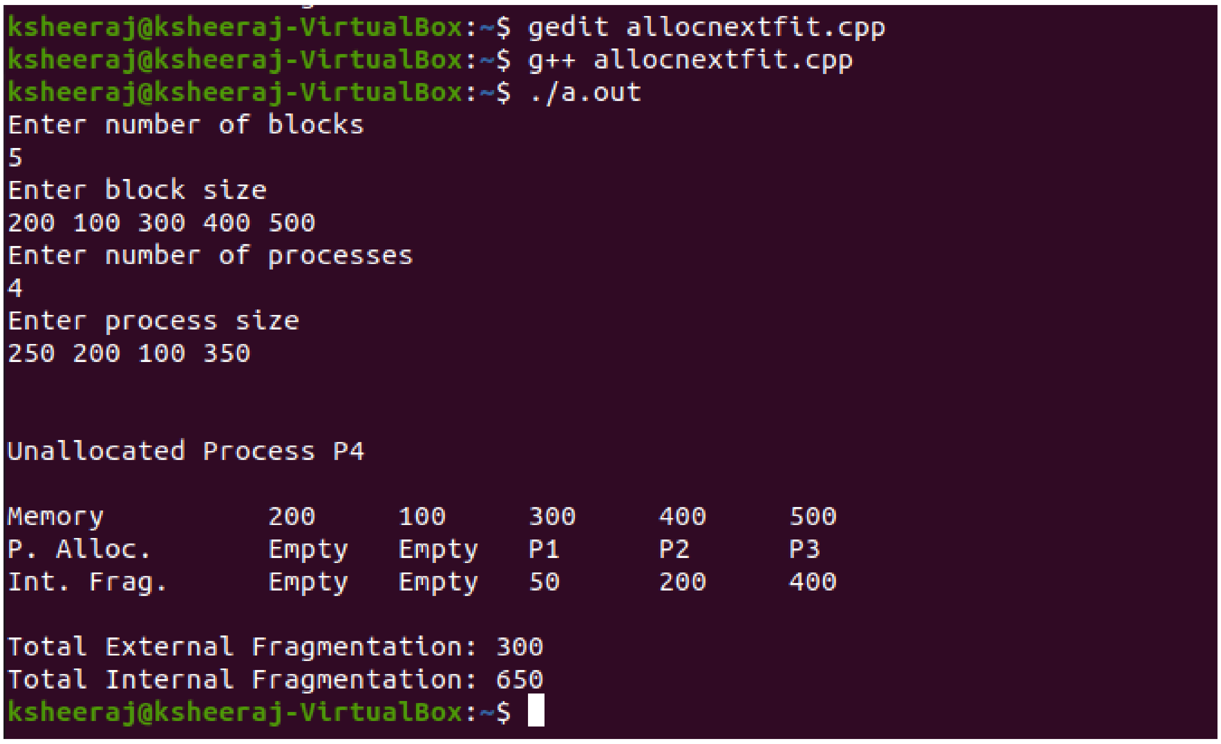


```
ksheeraj@ksheeraj-VirtualBox:~$ gedit allocworstfit.cpp
ksheeraj@ksheeraj-VirtualBox:~$ g++ allocworstfit.cpp
ksheeraj@ksheeraj-VirtualBox:~$ ./a.out
Enter number of blocks
5
Enter block size
500 400 300 200 100
Enter number of processes
4
Enter process size
90 200 280 300


Unallocated Process P4

Memory          500     400     300     200     100
P. Alloc.       P1      P2      P3      Empty   Empty
Int. Frag.      410     200     20      Empty   Empty

Total External Fragmentation: 300
Total Internal Fragmentation: 630
ksheeraj@ksheeraj-VirtualBox:~$
```

d) Next-Fit

```
#include<iostream>
#include<algorithm>
using namespace std;

struct node{
    int memsize;
    int allocp=-1;
    int pos;
    int allocSize;
}m[200];
```

```
bool posSort(node a,node b){
    return a.pos < b.pos;
}

bool memSort(node a,node b){
    return a.memsize < b.memsize;
}

int main(){
    int nm,np,choice, i, j, p[200];
    cout<<"Enter number of blocks\n";
    cin>>nm;
    cout<<"Enter block size\n";
    for(i=0;i<nm;i++){
        cin>>m[i].memsize;
        m[i].pos=i;
    }

    cout<<"Enter number of processes\n";
    cin>>np;

    cout<<"Enter process size\n";
    for(i=0;i<np;i++){
        cin>>p[i];
    }
    cout<<"\n\n";
    int globalFlag=0;
    int pos = -1;
    for(i=0;i<np;i++){
        int flag=0;
        for(j=pos+1;j<nm;j++){
            if(j==nm){
                j=0;

            }
            if(j==pos)
                break;
            if(p[i]<=m[j].memsize && m[j].allocp==-1){
                m[j].allocp=i;
                m[j].allocSize=p[i];
                flag=1;
                pos = j;
                if(j==nm-1){
                    j=0;
                    pos = -1;
                }
                break;
            }
        }
        if(flag==0){
                cout<<"Unallocated Process P"<<i+1<<"\n";
                globalFlag=1;
            }
        }
    sort(m,m+nm,posSort);
    cout<<"\n";
    int intFrag=0,extFrag=0;
    cout<<"Memory\t\t";
    for(i=0;i<nm;i++){
        cout<<m[i].memsize<<"\t";
    }
    cout<<"\n";
    cout<<"P. Alloc.\t";
    for(i=0;i<nm;i++){
        if(m[i].allocp!=-1){
            cout<<"P"<<m[i].allocp+1<<"\t";
        }
        else{
            cout<<"Empty\t";
        }
    }
    cout<<"\n";
    cout<<"Int. Frag.\t";
    for(i=0;i<nm;i++){
            if(m[i].allocp!=-1){
                cout<<m[i].memsize-m[i].allocSize<<"\t";
                intFrag+=m[i].memsize-m[i].allocSize;
            }
            else{
                extFrag+=m[i].memsize;
                cout<<"Empty\t";
            }
    }
    cout<<"\n";
    cout<<"\n";

    if(globalFlag==1)
        cout<<"Total External Fragmentation: 
"<<extFrag<<"\n";
    else
    {
        cout<<"Available Memory: "<<extFrag<<"\n";
    }

    cout<<"Total Internal Fragmentation: "<<intFrag<<"\n";


    return 0;
}
```



```
ksheeraj@ksheeraj-VirtualBox:~$ gedit allocnextfit.cpp
ksheeraj@ksheeraj-VirtualBox:~$ g++ allocnextfit.cpp
ksheeraj@ksheeraj-VirtualBox:~$ ./a.out
Enter number of blocks
5
Enter block size
200 100 300 400 500
Enter number of processes
4
Enter process size
250 200 100 350


Unallocated Process P4

Memory          200     100     300     400     500
P. Alloc.       Empty   Empty   P1      P2      P3
Int. Frag.      Empty   Empty   50      200     400

Total External Fragmentation: 300
Total Internal Fragmentation: 650
ksheeraj@ksheeraj-VirtualBox:~$
```