Experiment 8:

Code 1:

```
// Variable Partition : Contigous Allocation
//First Fit
#include<stdio.h>

int main()
{
    int noproc,blocks;

//---------------------------------
    printf("Enter the number of process : ");
    scanf("%d",&noproc);
    int proc_mem[noproc];


    printf("Enter the size of process P1 - Pn :\n");
    for(int i = 0;i<noproc;++i)
    {
    scanf("%d",&proc_mem[i]);
    }
//---------------------------------

    printf("Enter the number of blocks : ");
    scanf("%d",&blocks);
    int block_size[blocks];


    printf("Enter the size of blocks B1 - Bn :\n");
    for(int i = 0;i<noproc;++i)
    {
    scanf("%d",&block_size[i]);
    }
//---------------------------------
    for(int i = 0; i<noproc;++i)
    {
        for(int j = 0; j<blocks;j++)
        {
            if(proc_mem[i]<=block_size[j])
            {
                block_size[j] -= proc_mem[i];
                ++i;
```

```
            }
        }
    }
//---------------------------------
    int frag = 0;
    for(int i = 0;i<blocks;++i)
    {
        frag += block_size[i];
    }

    printf("Fragmentation is %d\n",frag);

    return 0;
}
```

Output 1:

```
Enter the number of process : 3
Enter the size of process P1 - Pn :
30
20
10
Enter the number of blocks : 3
Enter the size of blocks B1 - Bn :
20
10
30
Fragmentation is 20
```

Code 2:

```
// Variable Partition : Contigous Allocation
//Best Fit
#include<stdio.h>
int min_idx(int arr[],int size)
{
int min = 0;
    for(int i=0;i<size;++i)
    {
        if(arr[min]>arr[i])
        {
            min = i;
        }
    }
    return min;
}

//for max
```

```
int max_idx(int arr[],int size)
{
int min = 0;
    for(int i=0;i<size;++i)
    {
        if(arr[min]<arr[i])
        {
            min = i;
        }
    }
    return min;
}

int main()
{
    int noproc,blocks;

//---------------------------------
    printf("Enter the number of process : ");
    scanf("%d",&noproc);
    int proc_mem[noproc];


    printf("Enter the size of process P1 - Pn :\n");
    for(int i = 0;i<noproc;++i)
    {
    scanf("%d",&proc_mem[i]);
    }
//---------------------------------

    printf("Enter the number of blocks : ");
    scanf("%d",&blocks);
    int block_size[blocks];


    printf("Enter the size of blocks B1 - Bn :\n");
    for(int i = 0;i<noproc;++i)
    {
    scanf("%d",&block_size[i]);
    }
//---------------------------------
    int k = 10000,idx;
    for(int j = 0; j<noproc;j++)
    {
        int k = 100000,idx;

        for(int i = 0; i<blocks;++i)
        {
```

```
            if((block_size[i] - proc_mem[j])<k && block_size[i]>proc_mem[j])
            {
                k = block_size[i] - proc_mem[j];
                idx = i;
            }
        }
        block_size[idx] -= proc_mem[j];
    }
//---------------------------------
    int frag = 0;
    for(int i = 0;i<blocks;++i)
    {
        frag += block_size[i];
    }

    printf("Fragmentation is %d\n",frag);

    return 0;
}
```

Output 2:

```
Enter the number of process : 3
Enter the size of process P1 - Pn :
10
30
20
Enter the number of blocks : 3
Enter the size of blocks B1 - Bn :
10
20
30
Fragmentation is 0
```

Code 3:

```
// Variable Partition : Contigous Allocation
//Worst Fit
#include<stdio.h>
int min_idx(int arr[],int size)
{
int min = 0;
    for(int i=0;i<size;++i)
    {
        if(arr[min]>arr[i])
        {
```

```
            min = i;
        }
    }
    return min;
}

//for max
int max_idx(int arr[],int size)
{
int min = 0;
    for(int i=0;i<size;++i)
    {
        if(arr[min]<arr[i])
        {
            min = i;
        }
    }
    return min;
}

int main()
{
    int noproc,blocks;

//---------------------------------
    printf("Enter the number of process : ");
    scanf("%d",&noproc);
    int proc_mem[noproc];


    printf("Enter the size of process P1 - Pn :\n");
    for(int i = 0;i<noproc;++i)
    {
    scanf("%d",&proc_mem[i]);
    }
//---------------------------------

    printf("Enter the number of blocks : ");
    scanf("%d",&blocks);
    int block_size[blocks];


    printf("Enter the size of blocks B1 - Bn :\n");
    for(int i = 0;i<noproc;++i)
    {
    scanf("%d",&block_size[i]);
    }
//---------------------------------
```

```
    int k = 0,idx;
    for(int j = 0; j<noproc;j++)
    {
        int k = 0,idx;

        for(int i = 0; i<blocks;++i)
        {
            if(block_size[i]>=proc_mem[j] && block_size[i]>=0 &&
proc_mem[j]>=0 && (block_size[i] - proc_mem[j])>=1)
            {
                k = block_size[i] - proc_mem[j];
                idx = i;
            }
        }
        block_size[idx] -= proc_mem[j];
    }
//---------------------------------
    int frag = 0;
    for(int i = 0;i<blocks;++i)
    {
        printf("B%d : %d\n",i,block_size[i]);
        frag += block_size[i];
    }

    printf("Fragmentation is %d\n",frag);

    return 0;
}
```

Output 3:

```
Enter the number of process : 3
Enter the size of process P1 - Pn :
30
20
10
Enter the number of blocks : 3
Enter the size of blocks B1 - Bn :
20
10
30
Fragmentation is 20
```