WEEK6

M.No	Date	IDX	PROGRAM	Sign & Marks
6		Memory management		
		6.1	Implement the **FirstFit** memory allocation technique for fixed partition	
		6.2	Implement the **WorstFit** memory allocation technique for fixed partition.	
		6.3	Implement the **BestFit** memory allocation Technique for fixed partition.	
		6.4	Simulate Paging Technique of memory management.	

6.1

AIM : Write a program to Implement the **FirstFit** memory allocation methods for fixed partition.

PROGRAM :

```
#include<stdio.h>
#include<conio.h>
#define max 25
int main()
{
   int frag[max],b[max],f[max],i,j,nb,np,allocation[max],flag[max];
   printf("\nEnter the number of blocks:");
   scanf("%d",&nb);
   printf("Enter the number of Processes:");
   scanf("%d",&np);
   printf("\nEnter the size of the blocks:-\n");
   for(i=1;i<=nb;i++)
   {
      printf("Block %d:",i);
      scanf("%d",&b[i]);
   }
   printf("Enter the size of the processes :-\n");
   for(i=1;i<=np;i++)
   {
      printf("Process %d:",i);
      scanf("%d",&f[i]);
   }

   for(i=1;i<=nb;i++)
      flag[i]=-1;
   for(i=1;i<=np;i++)
```

```
        allocation[i]=-1;

    for(i=1;i<=np;i++)
    {
        for(j=1;j<=nb;j++)
        {
            if(flag[j] ==-1)
            {
                if(b[j]>=f[i])
                {
                    allocation[i]=j;
                    frag[i]=b[j]-f[i];
                    flag[j]=1;
                    break;
                }
            }
        }
    }
    printf("\nPID\tPSize\tBNO\tBSize\tFragment");
    for(i=1;i<=np;i++)
    {
        printf("\n%3d\t%5d",i,f[i]);
        if(allocation[i] !=-1)
            printf("%5d\t%5d\t%d\n",allocation[i],b[allocation[i]],frag[i]);
        else
            printf(" Not Allocated\n");
    }
   return 0;
}
```

OUTPUT :

```
Enter the number of blocks:5
Enter the number of Processes:4

Enter the size of the blocks:-
Block 1:100
Block 2:500
Block 3:200
Block 4:300
Block 5:600
Enter the size of the processes :-
Process 1:212
Process 2:417
Process 3:112
Process 4:426

PID     PSize   BNO     BSize   Fragment
  1       212    2        500   288

  2       417    5        600   183

  3       112    3        200   88

  4       426 Not Allocated
```

6.2

AIM : Implement the **WorstFit** memory allocation Technique for fixed partition.

PROGRAM :

```
#include<stdio.h>
#include<conio.h>
#define max 25
int main()
{
   int widx,frag[max],b[max],f[max],i,j,nb,np,allocation[max],flag[max];
   printf("\nEnter the number of blocks:");
   scanf("%d",&nb);
   printf("Enter the number of Processes:");
   scanf("%d",&np);
   printf("\nEnter the size of the blocks:-\n");
   for(i=1;i<=nb;i++)
   {
      printf("Block %d:",i);
      scanf("%d",&b[i]);
   }
   printf("Enter the size of the processes :-\n");
   for(i=1;i<=np;i++)
   {
      printf("Process %d:",i);
      scanf("%d",&f[i]);
   }

   for(i=1;i<=nb;i++)
```

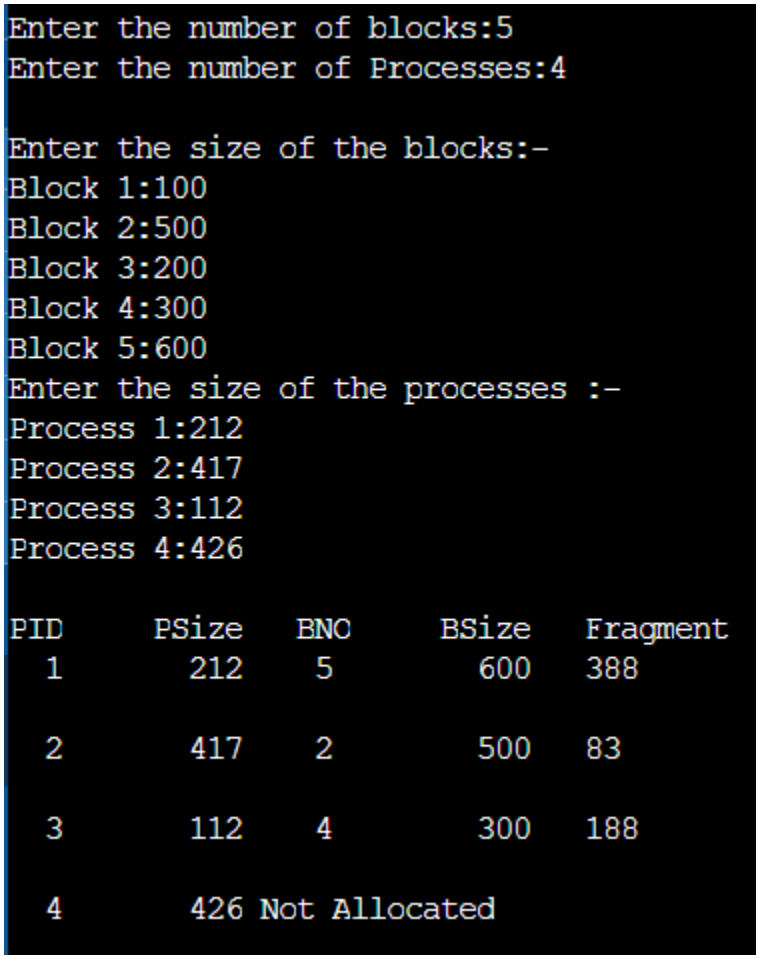
```
   flag[i]=-1;
for(i=1;i<=np;i++)
   allocation[i]=-1;

for(i=1;i<=np;i++)
{
   widx=-1;
   for(j=1;j<=nb;j++)
   {
      if(flag[j] ==-1)
      {
         if(b[j]>=f[i])
         {
            if(widx == -1)
              widx=j;
            else if(b[widx]<b[j])
              widx=j;
         }
      }
   }
   if(widx !=-1)
   {
      allocation[i]=widx;
      flag[widx]=1;
      frag[i]=b[widx]-f[i];
   }
}

printf("\nPID\tPSize\tBNO\tBSize\tFragment");
for(i=1;i<=np;i++)
{
   printf("\n%3d\t%5d",i,f[i]);
   if(allocation[i] !=-1)
```

```
            printf("%5d\t%5d\t%d\n",allocation[i],b[allocation[i]],frag[i]);
        else
            printf(" Not Allocated\n");
    }
   return 0;
}
```

OUTPUT :



6.3

AIM : Implement the **BestFit** memory allocation Technique for fixed partition.

PROGRAM :

```
#include<stdio.h>
#include<conio.h>
#define max 25
int main()
{
   int bidx,frag[max],b[max],f[max],i,j,nb,np,allocation[max],flag[max];
   printf("\nEnter the number of blocks:");
   scanf("%d",&nb);
   printf("Enter the number of Processes:");
   scanf("%d",&np);
   printf("\nEnter the size of the blocks:-\n");
   for(i=1;i<=nb;i++)
   {
      printf("Block %d:",i);
      scanf("%d",&b[i]);
   }
   printf("Enter the size of the processes :-\n");
   for(i=1;i<=np;i++)
   {
      printf("Process %d:",i);
      scanf("%d",&f[i]);
   }

   for(i=1;i<=nb;i++)
      flag[i]=-1;
   for(i=1;i<=np;i++)
      allocation[i]=-1;

   for(i=1;i<=np;i++)
```

```
    {
        bidx=-1;
        for(j=1;j<=nb;j++)
        {
            if(flag[j] ==-1)
            {
                if(b[j]>=f[i])
                {
                    if(bidx == -1)
                      bidx=j;
                    else if(b[bidx]>b[j])
                      bidx=j;
                }
            }
        }
        if(bidx !=-1)
        {
            allocation[i]=bidx;
            flag[bidx]=1;
            frag[i]=b[bidx]-f[i];
        }
    }
        printf("\nPID\tPSize\tBNO\tBSize\tFragment");
    for(i=1;i<=np;i++)
    {
        printf("\n%3d\t%5d",i,f[i]);
        if(allocation[i] !=-1)
            printf("%5d\t%5d\t%d\n",allocation[i],b[allocation[i]],frag[i]);
        else
            printf(" Not Allocated\n");
    }
   return 0;
}
```

OUTPUT :

```
Enter the number of blocks:5
Enter the number of Processes:4

Enter the size of the blocks:-
Block 1:100
Block 2:500
Block 3:200
Block 4:300
Block 5:600
Enter the size of the processes :-
Process 1:212
Process 2:417
Process 3:112
Process 4:426

PID     PSize   BNO     BSize   Fragment
 1       212     4       300    88

 2       417     2       500    83

 3       112     3       200    88

 4       426     5       600    174
```

6.4

AIM : Write a C program to simulate paging technique of memory management

PROGRAM :

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int ms,size,nf,ps,np,i,rn,pt[10],list[10],flag,p,la,pa,offset;

        printf("\nEnter the Size of the Main Memory : ");
        scanf("%d",&ms);
        printf("\nEnter the size of the page or frame : ");
        scanf("%d",&size);
        nf=ms/size;
        printf("\nNo Of Available Frames = %d",nf);
        printf("\nEnter the size of Process : ");
        scanf("%d",&ps);
        np=ps/size;
        i=ps%size;
        if(i!=0)
          np=np+1;
        printf("\n No Of Pages required = %d",np);
        if(np>nf)
        {
          printf("\n Memory Not Sufficient to execute the process ");
                exit(0);
        }
        else
        {
```

```
        for(i=0; i<nf;i++)
          list[i]=-1;
        for(i=0; i<np; i++)
        {
         flag=1;
         while(flag)
         {
          rn= rand() % nf;
          if(list[rn]==-1)
          {
                list[rn]=rn;
                 pt[i]=rn;
                 flag=0;
          }
         }
        }
        printf("\n Contents of page table are ");

   for (i=0;i<np;i++)
   {
        printf("\n%d \t %d",i,pt[i]);
   }

   printf("\n Enter the logical address : ");
   scanf("%d",&la);
        if (la>=ps)
        {
printf(" \n You are trying to access some other user memory -- access
denied");
        }
        else
        {
   p=la/size;
```

```
    offset=la%size;
    pa=pt[p]*size+offset;
    printf("\n physical address = %d",pa);
        }
        }
        return 0;
}
```

OUTPUT :