PROGRAM 6

6. Develop a C program to simulate the following contiguous memory allocation Techniques: a) Worst fit b) Best fit c) First fit

DESCRIPTION:

One of the simplest methods for memory allocation is to divide memory into several fixed-sized partitions. Each partition may contain exactly one process. In this multiple-partition method, when a partition is free, a process is selected from the input queue and is loaded into the free partition. When the process terminates, the partition becomes available for another process. The operating system keeps a table indicating which parts of memory are available and which are occupied. Finally, when a process arrives and needs memory, a memory section large enough for this process is provided. When it is time to load or swap a process into main memory, and if there is more than one free block of memory of sufficient size, then the operating system must decide which free block to allocate.

Best-fit strategy chooses the block that is closest in size to the request.

First-fit chooses the first available block that is large enough.

Worst-fit chooses the largest available block.

PROGRAM

a) Worst fit

```
#include<stdio.h>
//#include<conio.h>
#define max 25
void main()
{
        int frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0;
        static int bf[max],ff[max];

        printf("\n\tMemory Management Scheme - Worst Fit");
        printf("\nEnter the number of blocks:");
        scanf("%d",&nb);
        printf("Enter the number of files:");
        scanf("%d",&nf);
        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 files :-\n");
```

```
        for(i=1;i<=nf;i++)
        {
                printf("File %d:",i);
                scanf("%d",&f[i]);
        }
        for(i=1;i<=nf;i++)
        {

                for(j=1;j<=nb;j++)
                {
                        if(bf[j]!=1) //if bf[j] is not allocated
                        {
                                temp=b[j]-f[i];
                                if(temp>=0)
                                        if(highest<temp)
                                        {
                                        ff[i]=j;
                                        highest=temp;
                                        }
                        }
                }
                frag[i]=highest;
                bf[ff[i]]=1;
                highest=0;
        }
        printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragement");
        for(i=1;i<=nf;i++)
        printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
        //getch();
}
```

OUTPUT

```
Memory Management Scheme - Worst Fit
Enter the number of blocks:5
Enter the number of files: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 files :-
File 1:212
File 2:417
File 3:112
File 4:426

File_no:    File_size :    Block_no:    Block_size:    Fragement
1           212            5            600            388
2           417            2            500            83
```

3	112	4	300	188
4	426	0	0	0

b) Best fit

```
#include<stdio.h>
//#include <conio.h>
#define max 25
void main()
{
	int frag[max],b[max],f[max],i,j,nb,nf,temp,lowest=10000;
	static int bf[max],ff[max];

	printf("\nEnter the number of blocks:");
	scanf("%d",&nb);
	printf("Enter the number of files:");
	scanf("%d",&nf);
	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 files :-\n");
	for(i=1;i<=nf;i++)
	{
		printf("File %d:",i);
		scanf("%d",&f[i]);
	}
	for(i=1;i<=nf;i++)
	{
		for(j=1;j<=nb;j++)
		{
			if(bf[j]!=1)
			{
				temp=b[j]-f[i];
				if(temp>=0)
				if(lowest>temp)
				{
				ff[i]=j;

				lowest=temp;
				}
			}
		}
		frag[i]=lowest;
		bf[ff[i]]=1;
		lowest=10000;
	}
	printf("\nFile No\tFile Size \tBlock No\tBlock Size\tFragment");
	for(i=1;i<=nf && ff[i]!=0;i++)
```

```
        printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);

}
```

OUTPUT

```
Enter the number of blocks:5
Enter the number of files: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 files :-
File 1:212
File 2:417
File 3:112
File 4:426
```

File No	File Size	Block No	Block Size	Fragment
1	212	4	300	88
2	417	2	500	83
3	112	3	200	88
4	426	5	600	174

c) First Fit

```
#include<stdio.h>
void main()
{
        int bsize[10], psize[10], bno, pno, flags[10], allocation[10], i, j;
        for(i = 0; i < 10; i++)
        {
                flags[i] = 0;
                allocation[i] = -1;
        }
        printf("Enter no. of blocks: ");
        scanf("%d", &bno);
        printf("Enter no. of processes: ");
        scanf("%d", &pno);

        printf("Enter size of each block:\n");
        for(i = 0; i < bno; i++)
        {
                printf("Block %d:",i);
                scanf("%d", &bsize[i]);
        }
        printf("\nEnter size of each process:\n");
        for(i = 0; i < pno; i++)
        {
                printf("Process %d:",i);
```

```
                scanf("%d", &psize[i]);
        }
        for(j = 0; j < bno; j++)        //allocation as per first fit
                for(i = 0; i < pno; i++)
                        if((flags[i] == 0) && (bsize[j] >= psize[i]))
                        {
                                allocation[i] = j;
                                flags[i] = 1;
                                break;
                        }
        //display allocation details
        printf("\nFile No\t\tFile size\tBlock no.\tBlock size\tFragment");
        for(i = 0; i < pno; i++)
        {
                printf("\n%d\t\t%d\t\t", i+1, psize[i]);
                if(flags[i] == 1)
                        printf("%d\t\t%d\t\t%d",allocation[i]+1,bsize[allocation[i]],
(bsize[allocation[i]]-psize[i]));
                else
                        printf("0\t\t0\t\t0");
        }
}
```

OUTPUT

```
Enter no. of blocks: 5
Enter no. of processes: 4
Enter size of each block:
Block 0:100
Block 1:500
Block 2:200
Block 3:300
Block 4:600

Enter size of each process:
Process 0:212
Process 1:417
Process 2:112
Process 3:426

File No         File size       Block no.       Block size      Fragment
1               212             2               500             288
2               417             5               600             183
3               112             3               200             88
4               426             0               0               0
```