## WEEK 9

**Write a C program to simulate the following contiguous memory allocation techniques**

**a) Worst-fit**

**b) Best-fit**

**c) First-fit**

**CODE :**

**a. First fit**

```
#include <stdio.h>
#include <conio.h>
#define max 25
void main()
{
   int frag[max], b[max], f[max], i, j, nb, nf, temp;
   int bf[max], ff[max];
   printf("\n\tMemory Management Scheme - First 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++)
{
	temp = -1; // Reset temp to -1 for each new file
	for (j = 1; j <= nb; j++)
	{
	if (bf[j] != 1)
	{
		if (b[j] >= f[i])
		{
		ff[i] = j;
		temp = b[j] - f[i];
		break;
		}
	}
	 }
	frag[i] = temp;
	if (temp != -1)
	{
	bf[ff[i]] = 1;
	}
```

```
    }
    printf("\nFile_no:\tFile_size:\tBlock_no:\tBlock_size:\tFragment");
    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();
}
```

## a )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++)
{
	lowest = 10000; // Reset lowest to a high value for each new file
	for (j = 1; j <= nb; j++)
	{
	if (bf[j] != 1)
	{
		temp = b[j] - f[i];
		if (temp >= 0 && lowest > temp)
	   {
		ff[i] = j;
		lowest = temp;
		}
	}
	}
	frag[i] = lowest;
	bf[ff[i]] = 1;
}
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]);
    }
  }
```

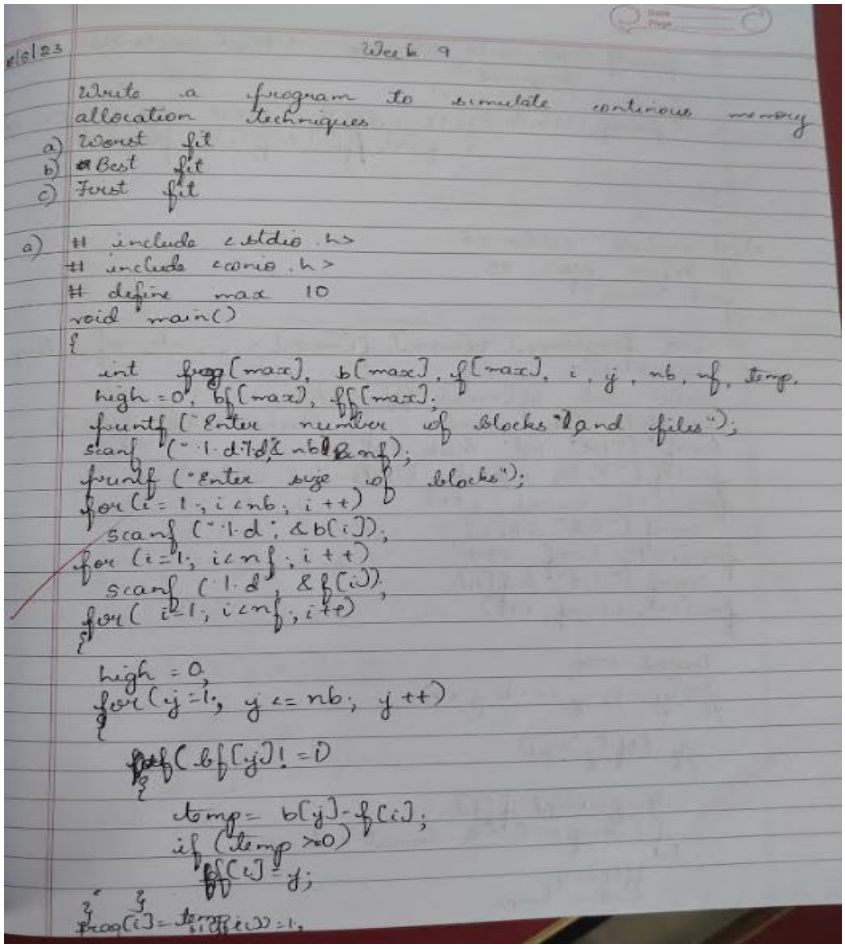
## a. Worst fit

```
#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;
    int bf[max], ff[max]; // Initialized these arrays to 0
    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++)
{
        highest = 0; // Reset highest to 0 for each new file
        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;
}
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]);
}
```

OBSERVATION :

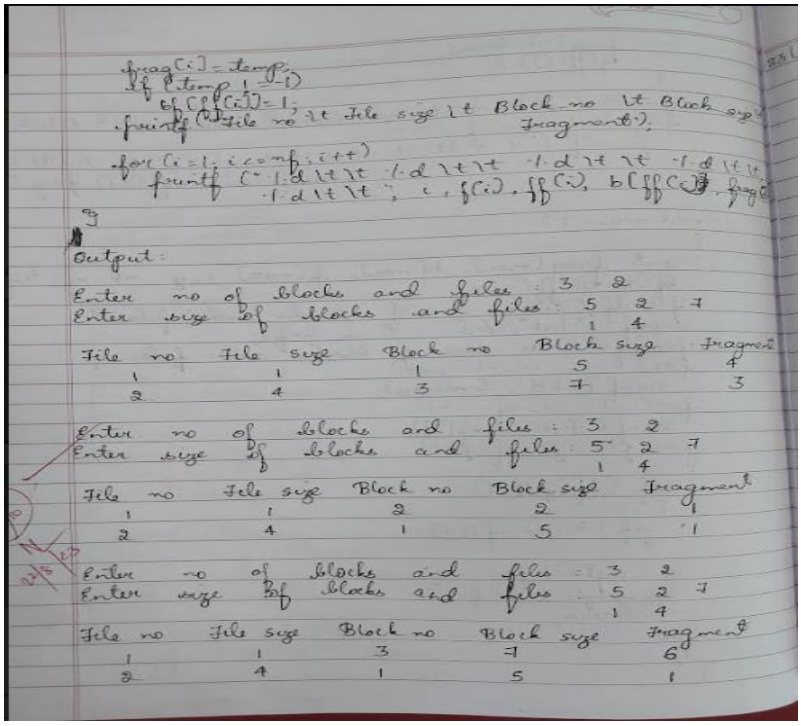


```
fprintf ("File no \t File size \t Block no \t Block size \t
        Fragment");
for (i=1; i<=nf; i++)
  fprintf ("\n %d \t\t %d \t\t %d \t\t %d \t\t %d", i, f[i], ff[i], b[ff[i]], frag[i]);
}
```

b)
```
#include <stdio.h>
#define MAX 25
void main()
{
  int frag[max], b[max], f[max], i, j, nb, nf, temp,
  lowest=1000;
  static int bf[max], ff[max];
  fprintf ("Enter number of blocks and files");
  scanf ("%d %d", &nb, &nf);
  fprintf ("Enter size of blocks and files");
  for (i=1; i<=nb; i++)
    scanf ("%d", &b[i]);
  for (i=1; i<=nf; i++)
    scanf ("%d", &f[i]);
  for (i=1; i<=nf; i++)
  {
    lowest=1000;
    for (j=1; j<=nb; j++)
    {
      if (bf[j]!=1)
      {
        temp=b[j]-f[i];
        if (temp>=0 && lowest>temp)
        {
          ff[i]=j;
          lowest=temp;
```

```
        frag[i]=lowest;
        bf[ff[i]]=1;
      }
    }
    fprintf ("File no \t File size \t Block no \t Block size \t Fragment");
    for (i=1; i<=nf; i++)
      fprintf ("\n %d \t\t %d \t\t %d \t\t %d \t\t %d", i, f[i], ff[i], b[ff[i]], frag[i]);
  }
```

c)
```
void main()
{
  int frag[max], b[max], f[max], i, j, nb, nf, temp;
  bf[max], ff[max];
  fprintf ("Enter no. of blocks and files");
  scanf ("%d %d", &nb, &nf);
  fprintf ("Enter size of blocks and files");
  for (i=1; i<=nb; i++)
    scanf ("%d", &mb[i]);
  for (i=1; i<=nf; i++)
    scanf ("%d", &mf[i]);
  for (j=1; j<=nf; j++)
  {
    if (bf[j]!=1)
    {
      if (b[j]>=f[i])
      {
        ff[i]=j;
        temp=b[j]-f[i];
        break;
      }
    }
  }
```



OUTPUT :

```
        Memory Management Scheme - First Fit
Enter the number of blocks:3
Enter the number of files:2

Enter the size of the blocks:
Block 1:5
Block 2:2
Block 3:7
Enter the size of the files:
File 1:1
File 2:4

File_no:        File_size:      Block_no:       Block_size:     Fragment
1               1               1               5               4
2               4               3               7               3
```

```
Enter the number of blocks:3
Enter the number of files:2

Enter the size of the blocks:
Block 1:5
Block 2:2
Block 3:7
Enter the size of the files:
File 1:1
File 2:4

File No File Size       Block No        Block Size      Fragment
1               1               2               2               1
2               4               1               5               1
```

```
        Memory Management Scheme - Worst Fit
Enter the number of blocks:3
Enter the number of files:2

Enter the size of the blocks:
Block 1:5
Block 2:2
Block 3:7
Enter the size of the files:
File 1:1
File 2:4

File_no:        File_size:      Block_no:       Block_size:     Fragement
1               1               3               7               6
2               4               1               5               1
```