```
/*******************************************************************************
Name – Sejal Girish Mahajan
Div - B-05
Problem Statement :-Implementation of memory placement strategies - best fit,worst fit,first
fit,next fit.
*******************************************************************************/
import java.util.Scanner;

class Sizes
{
	int size = 0;
	int block_alloc = 0;
	int index = 0;
}

public class Fitter
{
	static void FirstFit(int blocks[], Sizes sizes[], int n, int m)
	{
		int rem_size[] = new int[n];
		for(int i = 0 ; i < n; i++ )
			rem_size[i] = blocks[i];
		int bloc_alloc[] = new int[n];
		for(int i = 0; i < m; i++)
		{
			for(int j = 0; j < n; j++)
			{
				if(sizes[i].size <= rem_size[j])
				{
					rem_size[j] -= sizes[i].size;
					sizes[i].block_alloc = blocks[j];
```

```
                                sizes[i].index = j;
                                break;
                        }
                }
        }
        System.out.print("\nProcess No.\tSizes\tBlock Size\tRemaining Size\n");
        for(int i = 0; i < m; i++)
        {

                System.out.printf("%6d%13d%10d",i+1,sizes[i].size,sizes[i].block_alloc);
                if(sizes[i].block_alloc == 0)
                        System.out.printf("\t\tNo Block Allocated\n");
                else
                        System.out.printf("%15d\n",rem_size[sizes[i].index]);
        }
}
static void BestFit(int blocks[], Sizes sizes[], int n, int m)
{
        int j,i;
        int rem_size[] = new int[n];
        for(i = 0 ; i < n; i++)
                rem_size[i] = blocks[i];
        int min_ind = -1, min = -1;
        for(i = 0; i < m; i++)
        {
                min = min_ind = -1;
                boolean flag = true;
                for(j = 0; j < n; j++)
                {
                        if(sizes[i].size <= rem_size[j])
                        {
```

```
                                        if(min > rem_size[j] - sizes[i].size || flag == true)
                                        {
                                                min = rem_size[j] - sizes[i].size;
                                                min_ind = j;
                                                flag = false;
                                        }
                                }
                        }
                        if(!(min < 0))
                        {
                                rem_size[min_ind] = rem_size[min_ind] - sizes[i].size;
                                sizes[i].block_alloc = blocks[min_ind];
                                sizes[i].index = min_ind;
                        }
                }

                System.out.print("\nProcess No.\tSizes\tBlock Size\tRemaining Size\n");
                for(i = 0; i < m; i++)
                {

                        System.out.printf("%6d%13d%10d",i+1,sizes[i].size,sizes[i].block_alloc);
                        if(sizes[i].block_alloc == 0)
                                System.out.printf("\t\tNo Block Allocated\n");
                        else
                                System.out.printf("%15d\n",rem_size[sizes[i].index]);
                }
        }

        static void WorstFit(int blocks[], Sizes sizes[], int n, int m)
        {
                int j,i;
```

```
int rem_size[] = new int[n];
for(i = 0 ; i < n; i++)
        rem_size[i] = blocks[i];
int max_ind  = -1, max = -1;
for(i = 0; i < m; i++)
{
        max = max_ind = -1;
        boolean flag = true;
        for(j = 0; j < n; j++)
        {
                if(sizes[i].size <= rem_size[j])
                {
                        if( max < rem_size[j] - sizes[i].size || flag == true )
                        {
                                max = rem_size[j] - sizes[i].size;
                                max_ind = j;
                                flag = false;
                        }
                }
        }
        if(!(max < 0))
        {
                rem_size[max_ind] = rem_size[max_ind] - sizes[i].size;
                sizes[i].block_alloc = blocks[max_ind];
                sizes[i].index = max_ind;

        }
}

System.out.print("\nProcess No.\tSizes\tBlock Size\tRemaining Size\n");
for(i = 0; i < m; i++)
```

```
		{

			System.out.printf("%6d%13d%10d",i+1,sizes[i].size,sizes[i].block_alloc);
			if(sizes[i].block_alloc == 0)
				System.out.printf("\t\tNo Block Allocated\n");
			else
				System.out.printf("%15d\n",rem_size[sizes[i].index]);
		}
	}

	static void NextFit(int blocks[], Sizes sizes[], int n, int m)
	{
		int i,j,count = 0;
		int rem_size[] = new int[n];
		for(i = 0 ; i < n; i++ )
			rem_size[i] = blocks[i];

		for(j = 0; j < n; j++)
		{
			if(sizes[0].size <= rem_size[j])
			{
				rem_size[j] -= sizes[0].size;
				sizes[0].block_alloc = blocks[j];
				sizes[0].index = j;
				break;
			}
		}

		for(i = 1; i < m; i++)
		{
			for(count = 0,j = j + 1; count < n ;j++)
```

```
                {
                        j = j % n;
                        if(sizes[i].size <= rem_size[j])
                        {
                                rem_size[j] -= sizes[i].size;
                                sizes[i].block_alloc = blocks[j];
                                sizes[i].index = j;
                                break;
                        }

                        count++;
                        System.out.println(j);
                }
        }

        System.out.print("\nProcess No.\tSizes\tBlock Size\tRemaining Size\n");
        for(i = 0; i < m; i++)
        {

                System.out.printf("%6d%13d%10d",i+1,sizes[i].size,sizes[i].block_alloc);
                if(sizes[i].block_alloc == 0)
                        System.out.printf("\t\tNo Block Allocated\n");
                else
                        System.out.printf("%15d\n",rem_size[sizes[i].index]);
        }


}

public static void main(String[] args)
{
```

```
Scanner sc = new Scanner(System.in);
System.out.print("Enter the Number of Blocks : ");
int n = sc.nextInt();
int blocks[] = new int[n];
for(int i = 0; i < n; i++)
{
        System.out.print("Enter " + (i + 1) + "th Block's Size : ");
        blocks[i] = sc.nextInt();
}

System.out.print("\nEnter the Number of Sizes : ");
int m = sc.nextInt();
Sizes sizes[] = new Sizes[m];
for(int i = 0; i < m; i++)
        sizes[i] = new Sizes();

for(int i = 0; i < m; i++)
{
        System.out.print("Enter " + (i + 1) + "th Size : ");
        sizes[i].size = sc.nextInt();
}
int ch;
do
{
        System.out.println("\n---------MENU---------");
        System.out.println("1.First Fit");
        System.out.println("2.Best Fit");
        System.out.println("3.Worst Fit");
        System.out.println("4.Next Fit");
        System.out.println("5.Exit");
        System.out.print("Enter your choice : ");
```

```
                ch = sc.nextInt();
                switch(ch)
                {
                        case 1:
                                FirstFit(blocks,sizes,n,m);
                                break;
                        case 2:
                                BestFit(blocks,sizes,n,m);
                                break;
                        case 3:
                                WorstFit(blocks,sizes,n,m);
                                break;
                        case 4:
                                NextFit(blocks,sizes,n,m);
                                break;
                        case 5:
                                System.out.println("GG");
                                break;
                        default:
                                System.out.println("Wrong choice entered!!");
                                break;
                }
        }while(ch != 5);




    }

}
```

/*

OUTPUT :-----

gescoe@gescoe-OptiPlex-3010:~/Desktop/TE_44_SPOS/Java$ javac Fitter.java

gescoe@gescoe-OptiPlex-3010:~/Desktop/TE_44_SPOS/Java$ java Fitter

Enter the Number of Blocks : 6

Enter 1th Block's Size : 300

Enter 2th Block's Size : 600

Enter 3th Block's Size : 350

Enter 4th Block's Size : 200

Enter 5th Block's Size : 750

Enter 6th Block's Size : 125

Enter the Number of Sizes : 5

Enter 1th Size : 115

Enter 2th Size : 500

Enter 3th Size : 358

Enter 4th Size : 200

Enter 5th Size : 375

---------MENU---------

1.First Fit

2.Best Fit

3.Worst Fit

4.Next Fit

5.Exit

Enter your choice : 1

Process No.	Sizes	Block Size	Remaining Size
1	115	300	185
2	500	600	100

3	358	750	17
4	200	350	150
5	375	750	17

---------MENU---------

1.First Fit

2.Best Fit

3.Worst Fit

4.Next Fit

5.Exit

Enter your choice : 2

Process No.	Sizes	Block Size	Remaining Size
1	115	125	10
2	500	600	100
3	358	750	17
4	200	200	0
5	375	750	17

---------MENU---------

1.First Fit

2.Best Fit

3.Worst Fit

4.Next Fit

5.Exit

Enter your choice : 3

Process No.	Sizes	Block Size	Remaining Size
1	115	750	135
2	500	750	135
3	358	600	242

```
    4       200     350         150
    5       375     750         135

---------MENU---------
1.First Fit
2.Best Fit
3.Worst Fit
4.Next Fit
5.Exit
Enter your choice : 4

Process No.     Sizes   Block Size      Remaining Size
    1       115     300         185
    2       500     600         100
    3       358     750          17
    4       200     350         150
    5       375     750          17

---------MENU---------
1.First Fit
2.Best Fit
3.Worst Fit
4.Next Fit
5.Exit
Enter your choice : 5
GG

*/
```