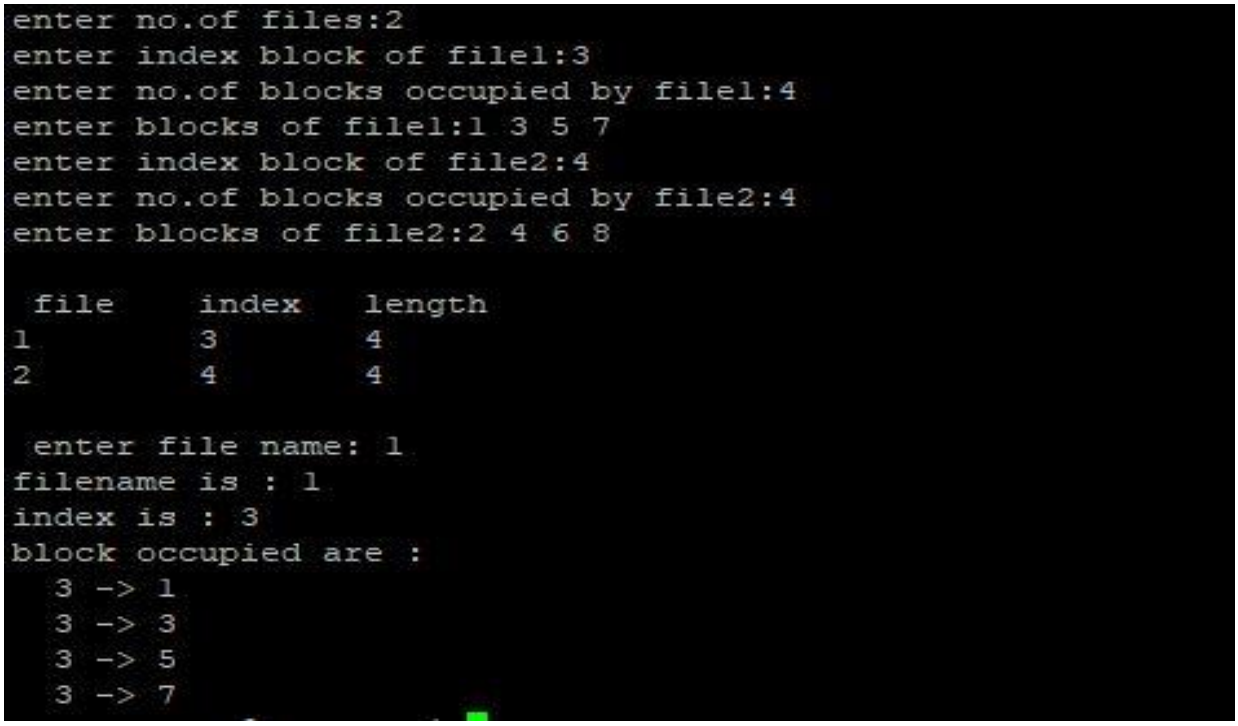
b)INDEXED FILE ALLOCATION STRATEGY

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
int main()
{
        int n,m[20],i,j,index[20],s[20],b[20][20],x;
        printf("enter no.of files:");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("enter index block of file%d:",i+1);
                scanf("%d",&index[i]);
                printf("enter no.of blocks occupied by file%d:",i+1);
                scanf("%d",&m[i]);
                printf("enter blocks of file%d:",i+1);
                for(j=0;j<m[i];j++)
                {
                        scanf("%d",&b[i][j]);
                }
        }
        printf("\n file \t index \t length \n");
        for(i=0;i<n;i++)
        {
                printf("%d \t %d \t %d \n",i+1,index[i],m[i]);
        }
        printf("\n enter file name: ");
        scanf("%d",&x);
        printf("filename is : %d \n",x);
```

```
        i=x-1;
        printf("index is : %d \n",index[i]);
        printf("block occupied are : \n");
        for(j=0;j<m[i];j++)
        {
                printf("%3d -> %d \n",index[i],b[i][j]);
        }
        return 0;
}
```

OUTPUT:



C)LINKED FILE ALLOCATION STRATEGY

PROGRAM:

```
#include<stdio.h>
#include<string.h>
struct file
{
	char fname[10];
	int start,size,block[10];
}f[10];
int main()
{
	int i,j,n;
	printf("enter no.of files:");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("enter file name:");
		scanf("%s",f[i].fname);
		printf("enter starting block:");
		scanf("%d",&f[i].start);
		f[i].block[0]=f[i].start;
		printf("enter no.of blocks:");
		scanf("%d",&f[i].size);
		printf("enter block numbers:");
		for(j=1;j<=f[i].size;j++)
		{
			scanf("%d",&f[i].block[j]);
		}
		printf("\n");
```

```
        }
        printf("file \t start \t size \t block \n");
        for(i=0;i<n;i++)
        {
                printf("%s \t %d \t %d \t",f[i].fname,f[i].start,f[i].size);
                for(j=1;j<=f[i].size-1;j++)
                {
                        printf("%d ---> ",f[i].block[j]);
                }
                printf("%d",f[i].block[j]);
                printf("\n");
        }
        return 0;
}
```

OUTPUT: