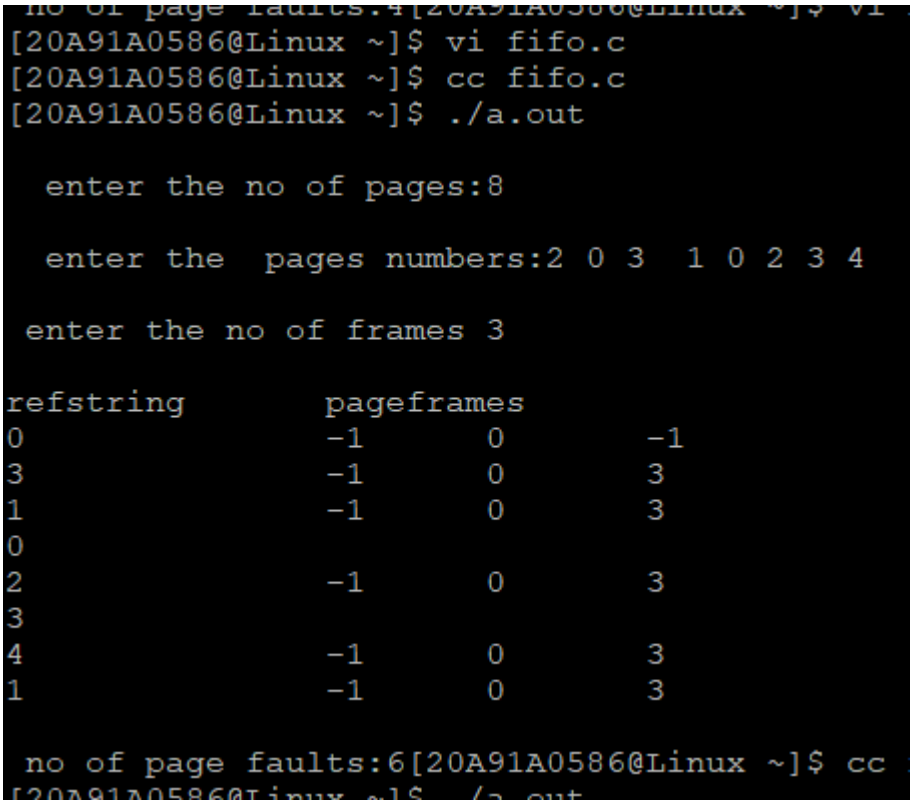
7. Simulate the FIFO page replacement algorithm

Program:

```
#include<stdio.h>
int main()
{
int i,j,n,a[50],frame[10],no,k,avail,count=0;
printf("\n  enter the no of pages:");
scanf("%d",&n);
printf("\n  enter the  pages numbers:");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\n enter the no of frames");
scanf("%d",&no);
for(i=0;i<no;i++)
frame[i]=-1;
j=0;
printf("\nrefstring\tpageframes\n");
for(i=1;i<=n;i++)
{
printf("%d\t\t",a[i]);
avail=0;
for(k=0;k<no;k++)
if(frame[k]==a[i])
avail=1;
if(avail==0)
```

```
{frame[i]=a[i];

j=(j+1)%no;

count++;

for(k=0;k<no;k++)

printf("%d\t",frame[k]);

}

printf("\n");

}

printf("\n no of page faults:%d",count);

return 0;

}
```

Output:



9. Simulate the 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;
pritnf(" enter no of files:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("enter filename:");
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 block :");
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\tstart\tsize\tblock\n");
for(i=0;i<n;i++)
{
printf("%S\t%d\t%d\n",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;
}
```

8. Simulate the LRU page replacement algorithm

Program:

```
#include<stdio.h>
int main()
{
int i,j,k, f,max,p=10,pf=0,count[10],pageref[25],fp[10],n,flag[10];
printf(" \n enter the length of page references string:");
scanf("%d",&n);
printf("\n enter the reference string--");
for(i=0;i<n;i++)
scanf("%d", &pageref[i]);
printf(" enter no of frames--");
scanf("%d",&f);
for(i=0;i<f;i++)
{
fp[i]=-1;count[i]=0;flag[i]=0;
}
printf(" the page replacement process is--");
for(i=0;i<n;i++)
{
for(k=0;k<f;k++)
{
if(count[k]==0)
{
fp[k]=pageref[i];
```

```
pf++;
count[k]=1;p=k;flag[k]=1;break;
}
elseif(fp[k]==pageref[i])
{
count[k]=1;p=k;flag[k]=1;break;
}
}
if(k==f)
{
max=0;
for(j=0;j<f;j++)
{
if(count[i]>max)
{
max=count[i];
p=j;
}
}
fp[p]=pageref[i];
count[p]=1;
flag[p]=1;
pf++;
}
printf(" pageref is %d",pageref[i]);
```

```
for(j=0;j<f;j++)
{
if(j==p||count[j]==0)
continue;
count[j]=count[j]+1;
}
for(j=0;j<f;j++)
{
printf("\t %d",fp[j]);
}
printf(" fault is %d",pf);
printf("\n");
printf(" the no of page faults using lru are %d",pf);
return 0;
}
```