Practical no: 12

Problem Statement: -To write a java program (using OOP feature) to implement LRU & Optimal algorithm for Page Replacement.

Name: Aditi Dinesh Mulay

Class: T.E. Computer

Subject: SPOS

Div: A

Roll no: 02

PRN No. 71918146B

SPOS

Assignment D-1

Aditi Dinesh Mulay
T.E. Comp D.v: A
Roll no:2

Aim: Implement page replacement algorithm.
1) LRU
2) Optimal.

Problem statement: To write a java program to implement LRU & Optimal algorithm for page replacement.

Prerequisites:- 1) Explain the concept virtual memory.
2) Define page replacement algorithm: LRU & Optimal.
3) Explain address translation in paging system.
4) Explain Belady's Anomaly.

Theory:

Whenever there is a page reference for which the page needed in memory, that event is called page fault or page failure situation. In such case we have to make space in memory for this new page by replacing any existing page. But we cannot replace any page We have to replace a page which is not used currently We can select appropriate page replacement policy. Designing appropriate algorithms to solve this problem is an important task because disk I/O is expensive.
1) Last recently used (LRU)
2) Optimal.

1) Last recently used page replacement.

The main difference between FIFO and optical page replacement is that the FIFO algorithms uses the time when the page was brought in to memory & the Optimal algorithms uses the time when a page is to be used. If we use the recent past as an approximation of the future then we will replace the page that has not been used for longest period of time. This approach is called as least recently used algorithm. LRU replacement associates with each page must be replaced. LRU chooses that page that has not been used for longest period of time. Now, consider reference string 7,0,1,2,0,3,4,2,3,03 with three memory frames or blocks. The first three frames reference cases page fault that fill the empty frames.

1	2	3	4	5	6	7	8	9	10	11	12
	7	0	1	2	0	3	0	4	2	3	0
	7	7	7	2	2	2	2	4	4	4	0
		0	0	0	0	0	0	0	0	3	3
		1	1	1	3	a	3	2	2	2	2
	+	+	+	+		+.		+			+

2) Optimal Page Replacement :

The algorithm has lowest page fault rate of all algorithm. It state that: Replace the page which will not be used for longest period of time. i.e. future knowledge of reference string is required.
→ Often called Balady's Min Basic idea: Replace the page that will not be refereneced for Longest time.

Impossible to implement.

Algorithm for LRU:

Let capacity be no. of pages that memory can hold. Let set be the current set of pages in memory:

1) Start traversing the pages.
i) if set holds less pages than capacity.
a) insert pages into the set one by one until the size of set reaches capacity or all pages requests processed.
b) Simultaneously, maintain recent occurred index of each page in a map called indexes.
c) increment page fault.
ii) Else.
If current page is present in set, do nothing.
Else
a) Find page in set that was least recently used. We find it using index array. We basically need to replace the page with minimum index.
b) Replace the found page with current page.
c) Increment page faults.
d) Update index of current page.

2) Return page faults.

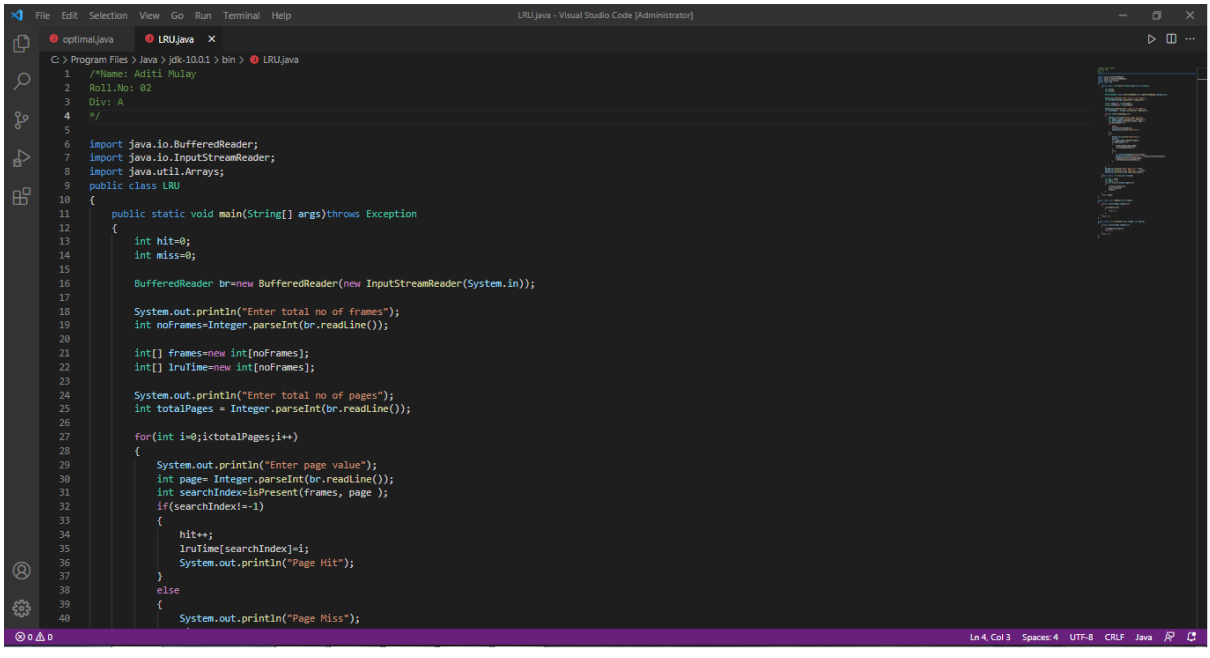
Algorithm for optimal:

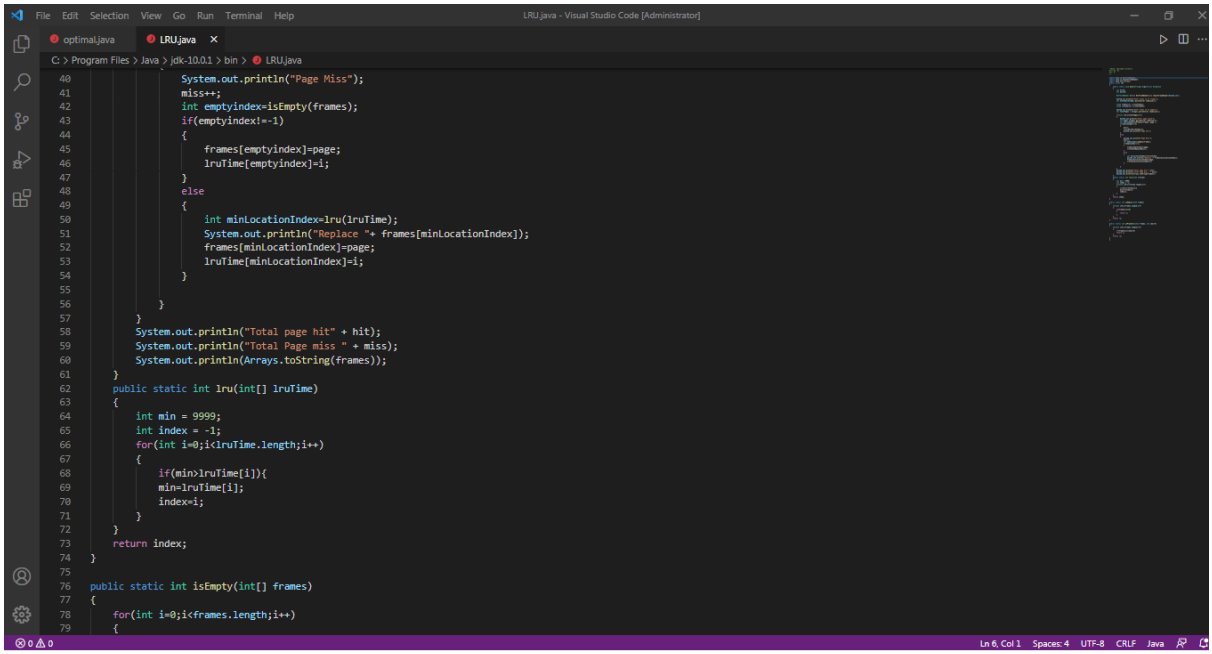
1. Start the process.
2. Declare the size.

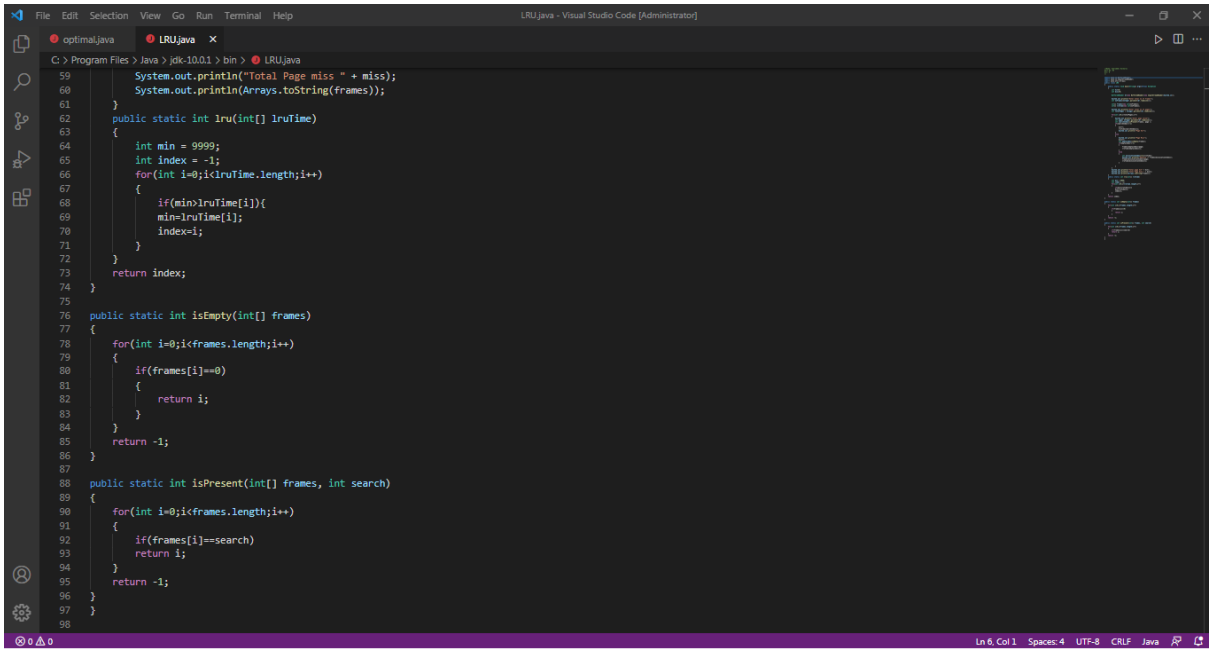
3. Get the value no. of pages to be inserted.
4. Get the value.
5. Compare counter label & stack.
6. Select the optimal page by counter value.
7. Stack them according the selection.
8. Print pages with fault pages.
9. Stop the process.

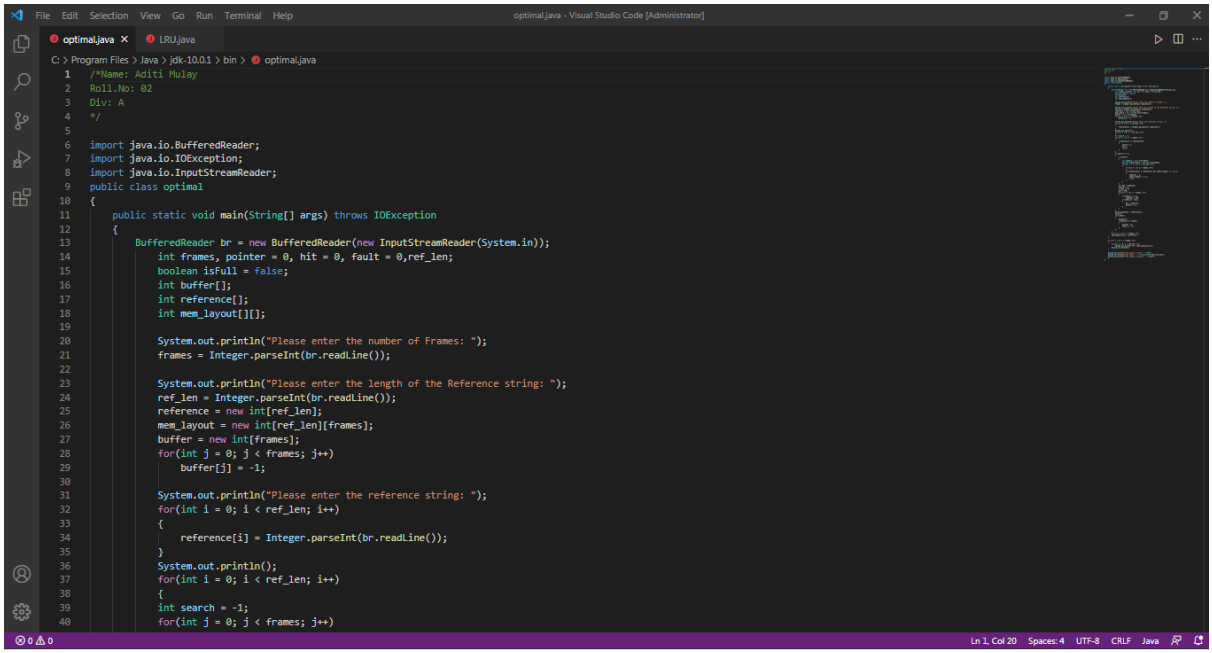
Conclusion:

Thus, we have successfuly implemented page replacement algorithm.

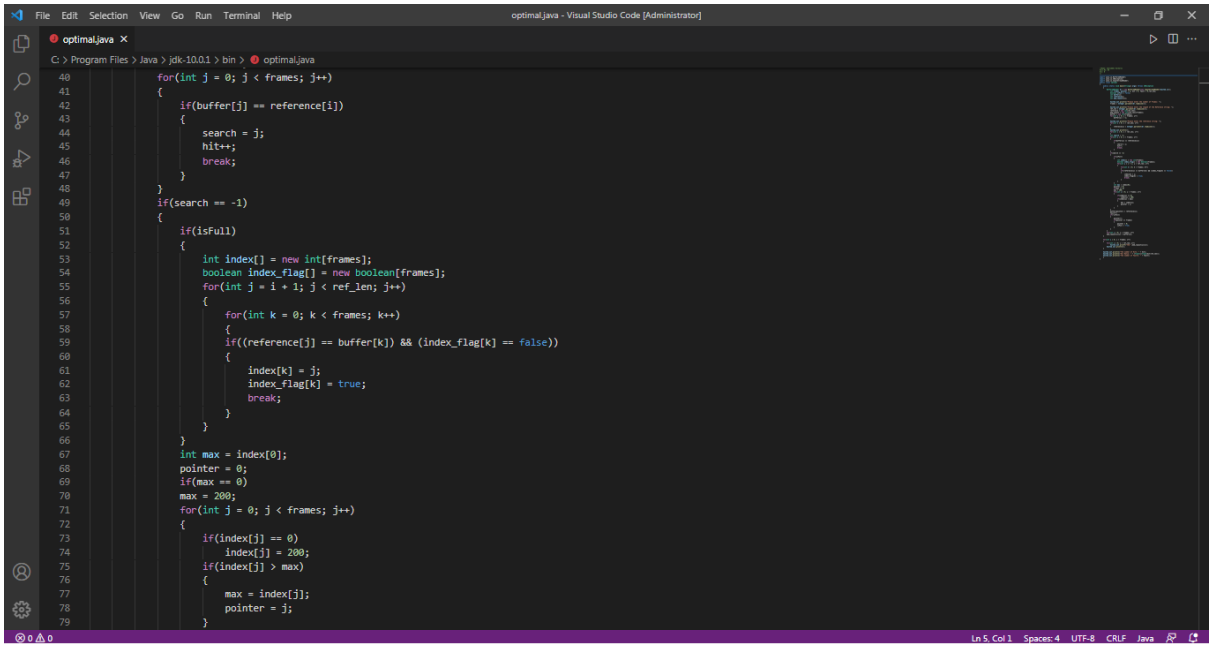



File Edit Selection View Go Run Terminal Help
LRU.java - Visual Studio Code [Administrator]
optimal.java
LRU.java
C: > Program Files > Java > jdk-10.0.1 > bin > LRU.java
System.out.println("Page Miss");
miss++;
int emptyindex=isEmpty(frames);
if(emptyindex!=-1)
{
frames[emptyindex]=page;
lruTime[emptyindex]=i;
}
else
{
int minLocationIndex=lru(lruTime);
System.out.println("Replace "+ frames[minLocationIndex]);
frames[minLocationIndex]=page;
lruTime[minLocationIndex]=i;
}
}
}
System.out.println("Total page hit" + hit);
System.out.println("Total Page miss " + miss);
System.out.println(Arrays.toString(frames));
}
public static int lru(int[] lruTime)
{
int min = 9999;
int index = -1;
for(int i=0;i<lruTime.length;i++)
{
if(min>lruTime[i]){
min=lruTime[i];
index=i;
}
}
return index;
}
public static int isEmpty(int[] frames)
{
for(int i=0;i<frames.length;i++)
{
Ln 6, Col 1 Spaces: 4 UTF-8 CRLF Java

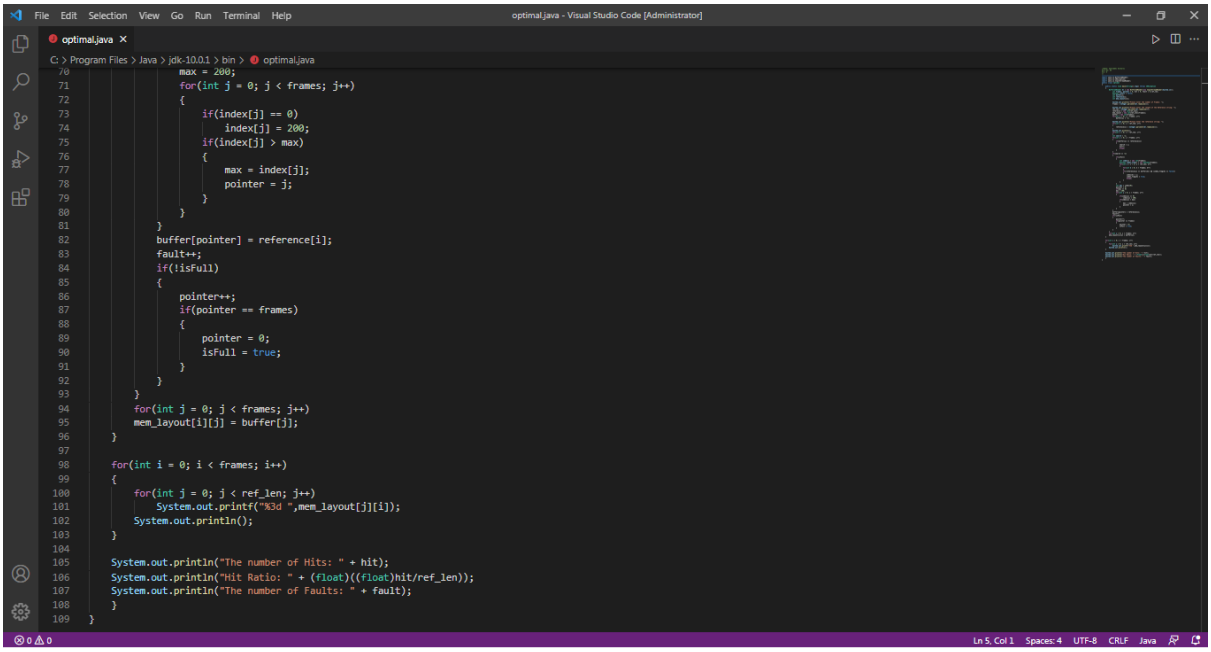

File Edit Selection View Go Run Terminal Help
LRU.java - Visual Studio Code [Administrator]
optimal.java
LRU.java
C: > Program Files > Java > jdk-10.0.1 > bin > LRU.java
System.out.println("Total Page miss " + miss);
System.out.println(Arrays.toString(frames));
}
public static int lru(int[] lruTime)
{
int min = 9999;
int index = -1;
for(int i=0;i<lruTime.length;i++)
{
if(min>lruTime[i]){
min=lruTime[i];
index=i;
}
}
return index;
}
public static int isEmpty(int[] frames)
{
for(int i=0;i<frames.length;i++)
{
if(frames[i]==0)
{
return i;
}
}
return -1;
}
public static int isPresent(int[] frames, int search)
{
for(int i=0;i<frames.length;i++)
{
if(frames[i]==search)
return i;
}
return -1;
}
}
Ln 6, Col 1 Spaces: 4 UTF-8 CRLF Java



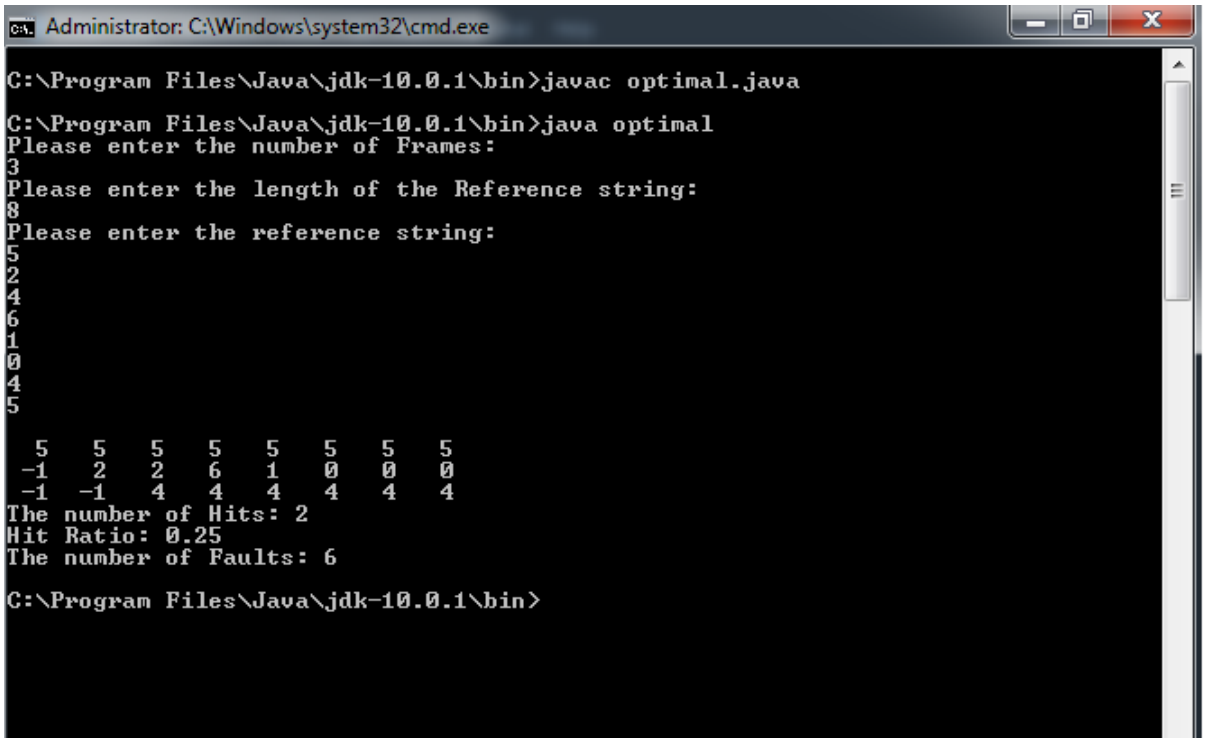
```java
/*Name: Aditi Mulay
Roll.No: 02
Div: A
*/

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class optimal
{
    public static void main(String[] args) throws IOException
    {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            int frames, pointer = 0, hit = 0, fault = 0,ref_len;
            boolean isFull = false;
            int buffer[];
            int reference[];
            int mem_layout[][];

            System.out.println("Please enter the number of Frames: ");
            frames = Integer.parseInt(br.readLine());

            System.out.println("Please enter the length of the Reference string: ");
            ref_len = Integer.parseInt(br.readLine());
            reference = new int[ref_len];
            mem_layout = new int[ref_len][frames];
            buffer = new int[frames];
            for(int j = 0; j < frames; j++)
                buffer[j] = -1;

            System.out.println("Please enter the reference string: ");
            for(int i = 0; i < ref_len; i++)
            {
                reference[i] = Integer.parseInt(br.readLine());
            }
            System.out.println();
            for(int i = 0; i < ref_len; i++)
            {
            int search = -1;
            for(int j = 0; j < frames; j++)
```



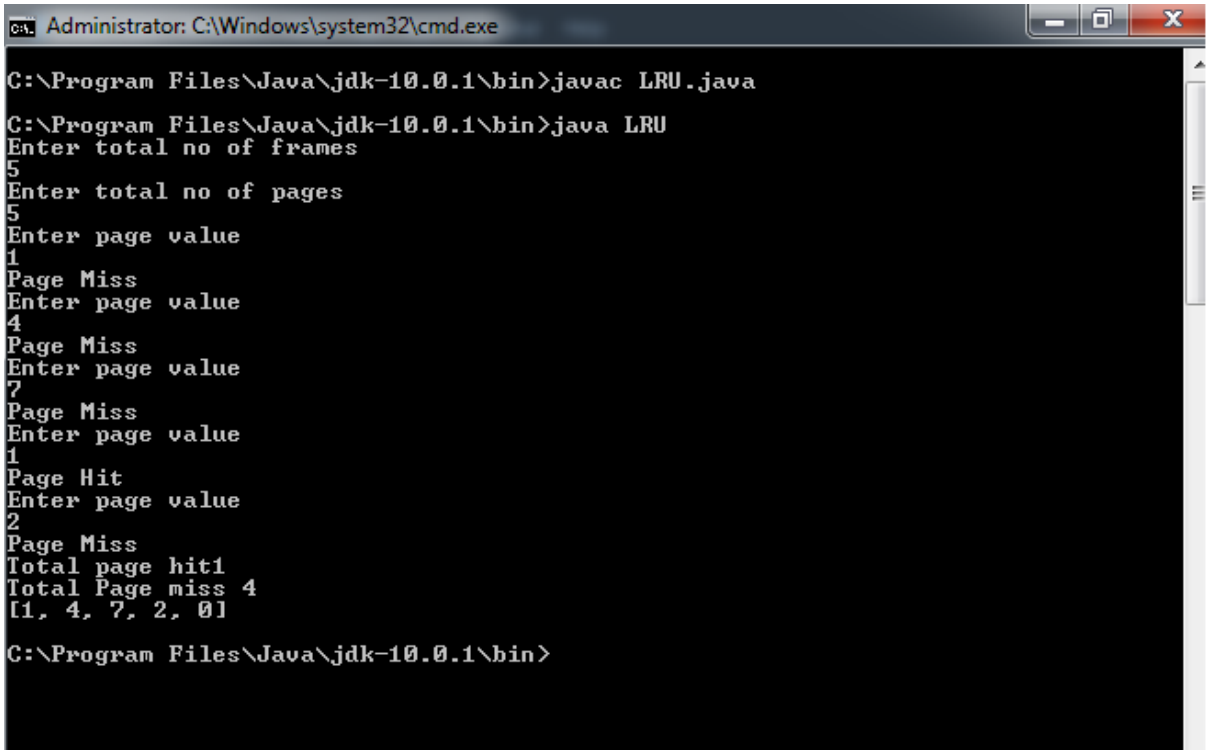
```java
            for(int j = 0; j < frames; j++)
            {
                if(buffer[j] == reference[i])
                {
                    search = j;
                    hit++;
                    break;
                }
            }
            if(search == -1)
            {
                if(isFull)
                {
                    int index[] = new int[frames];
                    boolean index_flag[] = new boolean[frames];
                    for(int j = i + 1; j < ref_len; j++)
                    {
                        for(int k = 0; k < frames; k++)
                        {
                        if((reference[j] == buffer[k]) && (index_flag[k] == false))
                        {
                            index[k] = j;
                            index_flag[k] = true;
                            break;
                        }
                    }
                }
                int max = index[0];
                pointer = 0;
                if(max == 0)
                max = 200;
                for(int j = 0; j < frames; j++)
                {
                    if(index[j] == 0)
                        index[j] = 200;
                    if(index[j] > max)
                    {
                        max = index[j];
                        pointer = j;
                    }
```



```
                max = 200;
                for(int j = 0; j < frames; j++)
                {
                    if(index[j] == 0)
                        index[j] = 200;
                    if(index[j] > max)
                    {
                        max = index[j];
                        pointer = j;
                    }
                }
            }
            buffer[pointer] = reference[i];
            fault++;
            if(!isFull)
            {
                pointer++;
                if(pointer == frames)
                {
                    pointer = 0;
                    isFull = true;
                }
            }
        }
        for(int j = 0; j < frames; j++)
        mem_layout[i][j] = buffer[j];
    }

    for(int i = 0; i < frames; i++)
    {
        for(int j = 0; j < ref_len; j++)
            System.out.printf("%3d ",mem_layout[j][i]);
        System.out.println();
    }

    System.out.println("The number of Hits: " + hit);
    System.out.println("Hit Ratio: " + (float)((float)hit/ref_len));
    System.out.println("The number of Faults: " + fault);
    }
}
```



```
Administrator: C:\Windows\system32\cmd.exe

C:\Program Files\Java\jdk-10.0.1\bin>javac optimal.java

C:\Program Files\Java\jdk-10.0.1\bin>java optimal
Please enter the number of Frames:
3
Please enter the length of the Reference string:
8
Please enter the reference string:
5
2
4
6
1
0
4
5

  5   5   5   5   5   5   5   5
 -1   2   2   6   1   0   0   0
 -1  -1   4   4   4   4   4   4
The number of Hits: 2
Hit Ratio: 0.25
The number of Faults: 6

C:\Program Files\Java\jdk-10.0.1\bin>
```


Administrator: C:\Windows\system32\cmd.exe
C:\Program Files\Java\jdk-10.0.1\bin>javac LRU.java
C:\Program Files\Java\jdk-10.0.1\bin>java LRU
Enter total no of frames
5
Enter total no of pages
5
Enter page value
1
Page Miss
Enter page value
4
Page Miss
Enter page value
7
Page Miss
Enter page value
1
Page Hit
Enter page value
2
Page Miss
Total page hit1
Total Page miss 4
[1, 4, 7, 2, 0]
C:\Program Files\Java\jdk-10.0.1\bin>