OS Lab Assignment 5

Krishna Pandey
U20CS110

Q1.) To implement first fit, best fit and worst fit storage allocation algorithms for memory Management. Also, calculate the total External Fragmentation after allocation.

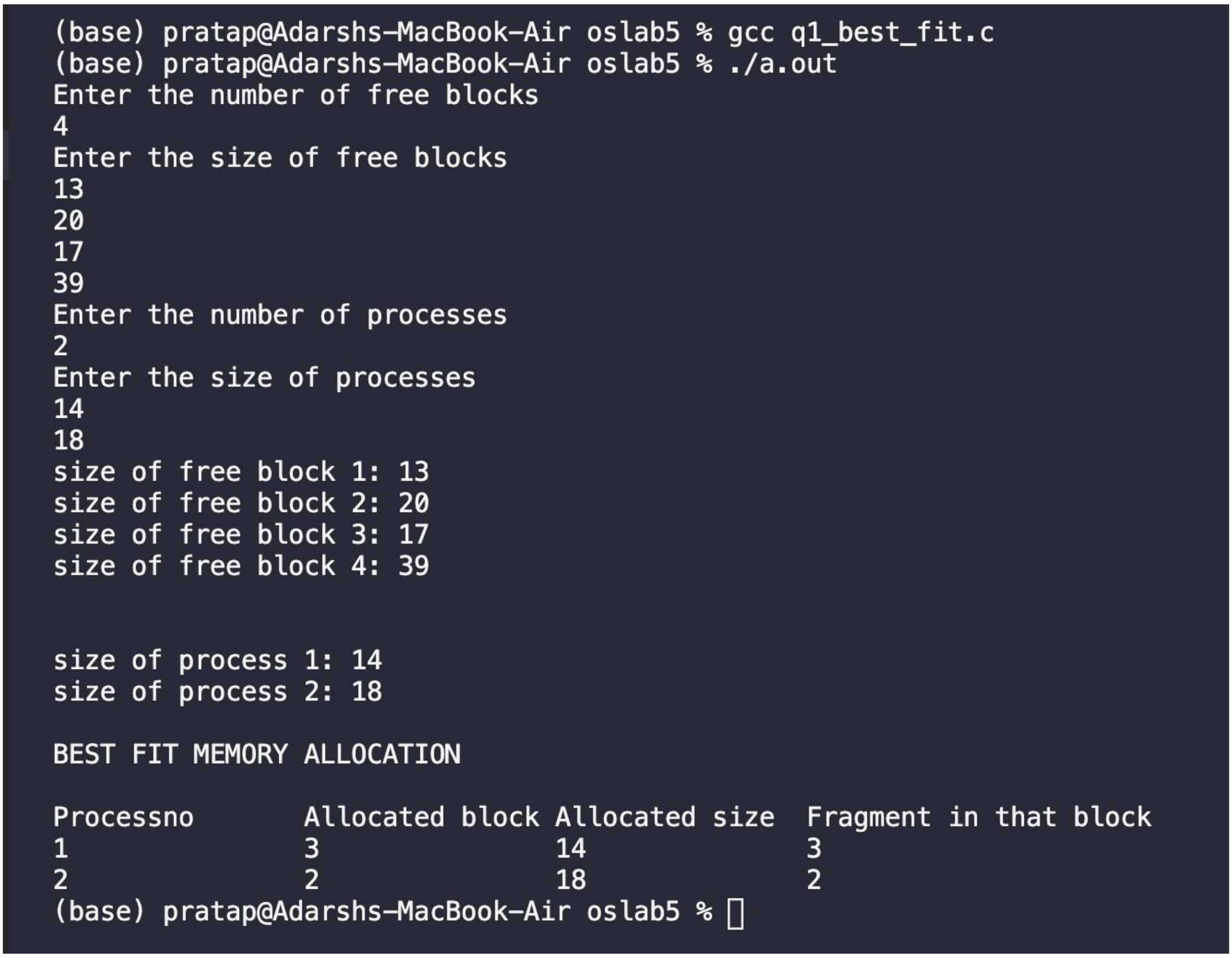
Best Fit

```
#include <stdio.h>
int main()
{
    int i, j, flag = 0, blockno, blocksize[10], processno,
processsize[10], lstsize, lstindex;

    printf("Enter the number of free blocks\n");
    scanf("%d", &blockno);

    printf("Enter the size of free blocks\n");
    for (i = 0; i < blockno; i++)
        scanf("%d", &blocksize[i]);

    printf("Enter the number of processes\n");
    scanf("%d", &processno);

    printf("Enter the size of processes\n");
    for (i = 0; i < processno; i++)
        scanf("%d", &processsize[i]);
    for (i = 0; i < blockno; i++)
        printf("size of free block %d: %d\n", i + 1,
blocksize[i]);
    printf("\n\n");
    for (i = 0; i < processno; i++)
        printf("size of process %d: %d\n", i + 1, processsize[i]);
    printf("\n");

    printf("BEST FIT MEMORY ALLOCATION\n\n");
    printf("Processno\tAllocated block\tAllocated size\tFragment
in that block\n");

    i = 0;
    while (i < processno)
    {
        flag = 0;
        for (j = 0; j < blockno; j++)
        {
            if (processsize[i] <= blocksize[j])
            {
```

```
if (flag == 0)
```

```
                {
                    lstsize = blocksize[j];
                    lstindex = j;
                    flag++;
                }
                else if (blocksize[j] < lstsize)
                {
                    lstsize = blocksize[j];
                    lstindex = j;
                }
            }
        }
        blocksize[lstindex] -= processsize[i];
        printf("%d\t\t%d\t\t%d\t\t%d\n", i + 1, lstindex + 1, 
processsize[i], blocksize[lstindex]);
        i++;
    }
    return 0;
}
```

Output

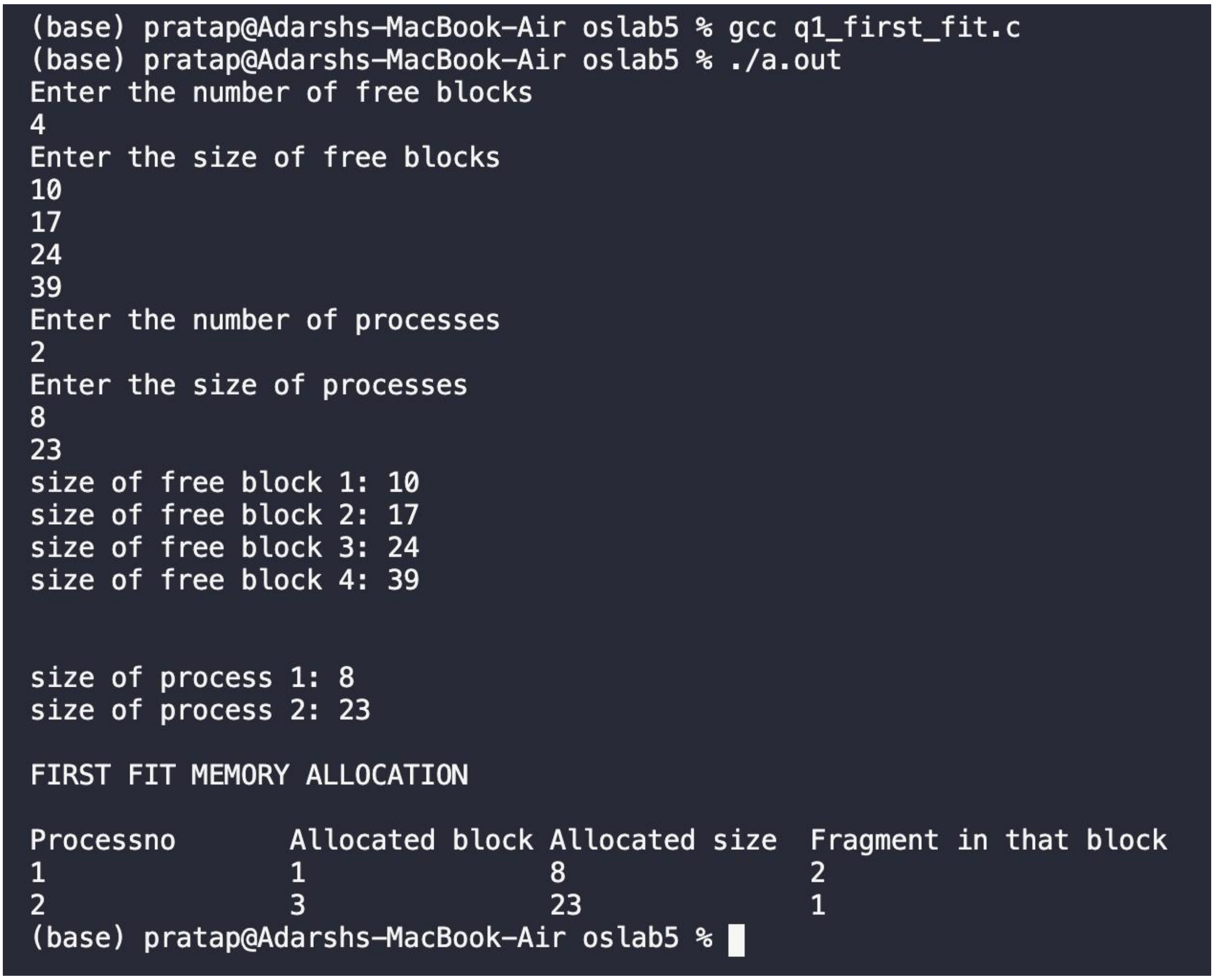


```
(base) pratap@Adarshs-MacBook-Air oslab5 % gcc q1_best_fit.c
(base) pratap@Adarshs-MacBook-Air oslab5 % ./a.out
Enter the number of free blocks
4
Enter the size of free blocks
13
20
17
39
Enter the number of processes
2
Enter the size of processes
14
18
size of free block 1: 13
size of free block 2: 20
size of free block 3: 17
size of free block 4: 39


size of process 1: 14
size of process 2: 18

BEST FIT MEMORY ALLOCATION

Processno       Allocated block Allocated size  Fragment in that block
1               3               14              3
2               2               18              2
(base) pratap@Adarshs-MacBook-Air oslab5 % 
```

First fit

```
#include <stdio.h>
void main()
{
    int i, j, blockno, blocksize[10], processno, processsize[10];
    printf("Enter the number of free blocks\n");
    scanf("%d", &blockno);
    printf("Enter the size of free blocks\n");
    for (i = 0; i < blockno; i++)
        scanf("%d", &blocksize[i]);
    printf("Enter the number of processes\n");
    scanf("%d", &processno);
    printf("Enter the size of processes\n");
    for (i = 0; i < processno; i++)
        scanf("%d", &processsize[i]);
    for (i = 0; i < blockno; i++)
        printf("size of free block %d: %d\n", i + 1,
blocksize[i]);
    printf("\n\n");
    for (i = 0; i < processno; i++)
        printf("size of process %d: %d\n", i + 1, processsize[i]);
    printf("\n");
    printf("FIRST FIT MEMORY ALLOCATION\n\n");
    printf("Processno\tAllocated block\tAllocated size\tFragment
in that block\n");
    i = 0;
    while (i < processno)
    {
        for (j = 0; j < blockno; j++)
        {
            if (processsize[i] <= blocksize[j])
            {
                blocksize[j] -= processsize[i];
                break;
            }
        }
        printf("%d\t\t%d\t\t%d\t\t%d\n", i + 1, j + 1,
processsize[i], blocksize[j]);
        i++;
    }
}
```

Output



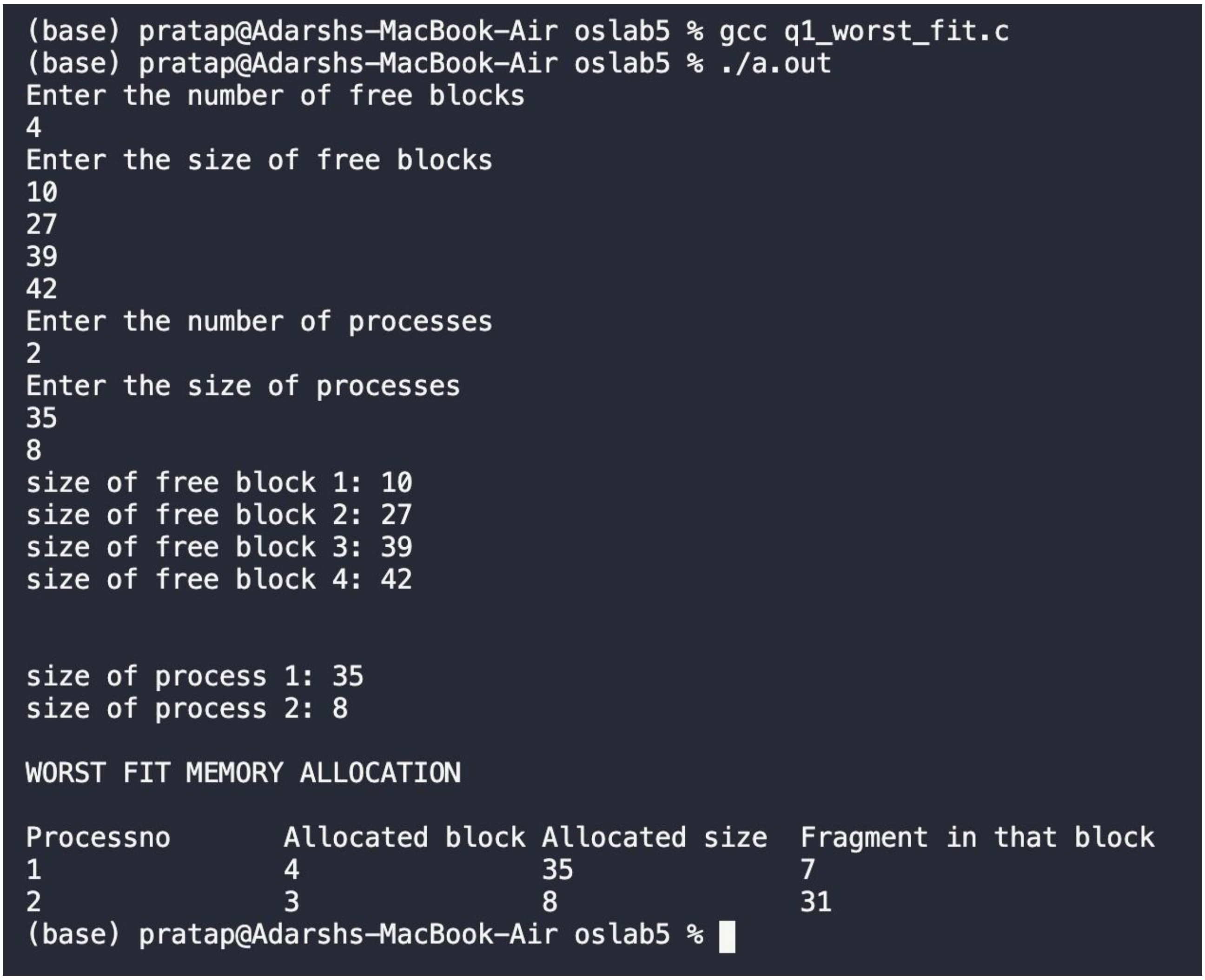
```
(base) pratap@Adarshs-MacBook-Air oslab5 % gcc q1_first_fit.c
(base) pratap@Adarshs-MacBook-Air oslab5 % ./a.out
Enter the number of free blocks
4
Enter the size of free blocks
10
17
24
39
Enter the number of processes
2
Enter the size of processes
8
23
size of free block 1: 10
size of free block 2: 17
size of free block 3: 24
size of free block 4: 39


size of process 1: 8
size of process 2: 23

FIRST FIT MEMORY ALLOCATION

Processno       Allocated block Allocated size  Fragment in that block
1               1               8               2
2               3               23              1
(base) pratap@Adarshs-MacBook-Air oslab5 %
```

Worst fit

```
#include <stdio.h>
int main()
{
    int i, j, flag = 0, blockno, blocksize[10], processno,
processsize[10], lstsize, lstindex;
    printf("Enter the number of free blocks\n");
    scanf("%d", &blockno);
    printf("Enter the size of free blocks\n");
    for (i = 0; i < blockno; i++)
        scanf("%d", &blocksize[i]);
    printf("Enter the number of processes\n");
    scanf("%d", &processno);
    printf("Enter the size of processes\n");
    for (i = 0; i < processno; i++)
        scanf("%d", &processsize[i]);
    for (i = 0; i < blockno; i++)
```

```
        printf("size of free block %d: %d\n", i + 1,
blocksize[i]);
    printf("\n\n");
    for (i = 0; i < processno; i++)
        printf("size of process %d: %d\n", i + 1, processsize[i]);
    printf("\n");
    printf("WORST FIT MEMORY ALLOCATION\n\n");
    printf("Processno\tAllocated block\tAllocated size\tFragment
in that block\n");
    i = 0;
    while (i < processno)
    {
        flag = 0;
        for (j = 0; j < blockno; j++)
        {
            if (processsize[i] <= blocksize[j])
            {
                if (flag == 0)
                {
                    lstsize = blocksize[j];
                    lstindex = j;
                    flag++;
                }
                else if (blocksize[j] > lstsize)
                {
                    lstsize = blocksize[j];
                    lstindex = j;
                }
            }
        }
        blocksize[lstindex] -= processsize[i];
        printf("%d\t\t%d\t\t%d\t\t%d\n", i + 1, lstindex + 1,
processsize[i], blocksize[lstindex]);
        i++;
    }
    return 0;
}
```

Output



```
(base) pratap@Adarshs-MacBook-Air oslab5 % gcc q1_worst_fit.c
(base) pratap@Adarshs-MacBook-Air oslab5 % ./a.out
Enter the number of free blocks
4
Enter the size of free blocks
10
27
39
42
Enter the number of processes
2
Enter the size of processes
35
8
size of free block 1: 10
size of free block 2: 27
size of free block 3: 39
size of free block 4: 42


size of process 1: 35
size of process 2: 8

WORST FIT MEMORY ALLOCATION

Processno       Allocated block Allocated size  Fragment in that block
1               4               35              7
2               3               8               31
(base) pratap@Adarshs-MacBook-Air oslab5 % 
```

Q2.)

Write a program that implements the following Page replacement algorithm.

i) LRU (Least Recently Used)

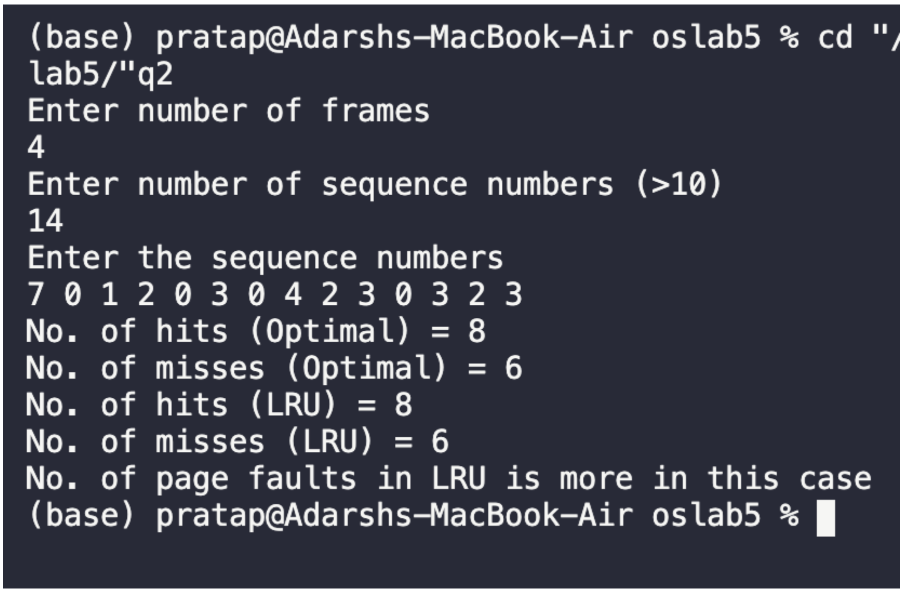
ii) Optimal Page Replacement algorithm

Count total numbers of page fault at the end and compare for both of the algorithms.

```
#include <bits/stdc++.h>
using namespace std;


int optimalMiss = 0;
// Function to check whether a page exists
// in a frame or not
bool search(int key, vector<int>& fr)
{
    for (int i = 0; i < fr.size(); i++)
```

```
        if (fr[i] == key)
            return true;
    return false;
}

// Function to find the frame that will not be used
// recently in future after given index in pg[0..pn-1]
int predict(int pg[], vector<int>& fr, int pn, int index)
{
    // Store the index of pages which are going
    // to be used recently in future
    int res = -1, farthest = index;
    for (int i = 0; i < fr.size(); i++) {
        int j;
        for (j = index; j < pn; j++) {
            if (fr[i] == pg[j]) {
                if (j > farthest) {
                    farthest = j;
                    res = i;
                }
                break;
            }
        }

        // If a page is never referenced in future,
        // return it.
        if (j == pn)
            return i;
    }

    // If all of the frames were not in future,
    // return any of them, we return 0. Otherwise
    // we return res.
    return (res == -1) ? 0 : res;
}

void optimalPage(int pg[], int pn, int fn)
{
    // Create an array for given number of
    // frames and initialize it as empty.
    vector<int> fr;

    // Traverse through page reference array
    // and check for miss and hit.
    int hit = 0;
    for (int i = 0; i < pn; i++) {

        // Page found in a frame : HIT
        if (search(pg[i], fr)) {
            hit++;
            continue;
        }
        // Page not found in a frame : MISS

        // If there is space available in frames.
        if (fr.size() < fn)
            fr.push_back(pg[i]);

        // Find the page to be replaced.
        else {
            int j = predict(pg, fr, pn, i + 1);
            fr[j] = pg[i];
        }
    }
    cout << "No. of hits (Optimal) = " << hit << endl;
    cout << "No. of misses (Optimal) = " << pn - hit << endl;
    optimalMiss = pn - hit;
}

int main()
{
    int fn;
    int n;
    cout << "Enter number of frames \n";
    cin >> fn;
    cout << "Enter number of sequence numbers (>10) \n";
    cin >> n;
    if (n<10)
    {
        cout << "Restart with more sequence numbers \n";
        return 0;
    }
    int pg[n];
    cout << "Enter the sequence numbers \n";
    int maxno = 0;
    for (int i = 0; i < n; i++)
    {
        cin >> pg[i];
        maxno = max(maxno,pg[i]);
    }
    if (fn >= maxno)
    {
        cout << "Total number of frames should be less than
highest sequence number \n";
        return 0;
    }
    optimalPage(pg, n, fn);
    deque<int> q(fn);
    int count = 0;
    int page_faults=0;
    deque<int>::iterator itr;
    q.clear();
    for(int j = 0; j < n; j++)
    {
        int i = pg[j];
        // Insert it into set if not present
        // already which represents page fault
        itr = find(q.begin(),q.end(),i);
        if(!(itr != q.end()))
        {
            ++page_faults;
            if(q.size() == fn)
            {
                q.erase(q.begin());
                q.push_back(i);
            }
            else
            {
                q.push_back(i);
            }
        }
        else
        {
        // Remove the indexes page
            q.erase(itr);
        // insert the current page
            q.push_back(i);
        }
    }
    cout << "No. of hits (LRU) = " << n - page_faults << endl;
    cout << "No. of misses (LRU) = " << page_faults << endl;
    if (optimalMiss > page_faults)
        cout << "No. of page faults in Optimal is more in this 
case \n";
    else
        cout << "No. of page faults in LRU is more in this case 
\n";
    return 0;
}
```

Output



```
(base) pratap@Adarshs-MacBook-Air oslab5 % cd "/
lab5/"q2
Enter number of frames
4
Enter number of sequence numbers (>10)
14
Enter the sequence numbers
7 0 1 2 0 3 0 4 2 3 0 3 2 3
No. of hits (Optimal) = 8
No. of misses (Optimal) = 6
No. of hits (LRU) = 8
No. of misses (LRU) = 6
No. of page faults in LRU is more in this case
(base) pratap@Adarshs-MacBook-Air oslab5 %
```