

Name: Lê Hoàng Khanh

ID: 20521448

Class: IT007.M12.MTCL.1

OPERATING SYSTEM LAB 6'S REPORT

SUMMARY

Task		Status	Page
Hướng dẫn thực hành	Sử dụng ngôn ngữ lập trình C viết chương trình mô phỏng các giải thuật thay thế trang đã nêu trong câu số 3 mục 6.3.3 với các yêu cầu.	Done	2
Bài tập ôn tập	1. Nghịch lý Belady là gì? Sử dụng chương trình đã viết trên để chứng minh nghịch lý này.	Done	14
	2. Nhận xét về mức độ hiệu quả và tính khả thi của các giải thuật FIFO, OPT, LRU.	Done	14

Self-scores:

Note: Export file to **PDF and name the file by following format:
LAB X – <Student ID>.pdf*

6.4/ Hướng dẫn thực hành

1. Sử dụng ngôn ngữ lập trình C viết chương trình mô phỏng các giải thuật thay thế trang đã nêu trong câu số 3 mục 6.3.3 với các yêu cầu.

Code:

Code

```
#include <iostream>
using namespace std;
static int preArray[10];
static int page_fault[100];
static int number_page_fault = 1;

int Is_in_preArray(int page, int value) {
    for (int i = 0; i < page; i++) {
        if (value == preArray[i]) return i;
    }
    return -1;
}

int Farest_Element(int *ref, int pivot, int page) {
    int min = pivot;
    int vt;
    for (int i = 0; i < page; i++) {
        for (int j = pivot - 1; j >= 0; j--) {
            if (preArray[i] == ref[j]) {
                if (j < min) {
                    min = j;
                    vt = i;
                }
            }
        }
    }
}
```

```

        break;
    }

}

}
return vt;
}

int Fareast_Element_Oppsite(int *ref, int pivot, int page, int n) {
    int max = pivot;
    int flag[10];
    for (int i = 0; i < page; i++) {
        flag[i] = false;
    }
    int vt = -1;
    for (int i = 0; i < page; i++) {
        for (int j = pivot + 1; j < n; j++) {
            if (preArray[i] == ref[j]) {
                if (j > max) {
                    max = j;
                    vt = i;
                }
                flag[i] = true;
                break;
            }
        }
    }
}

```

```

    for (int i = 0; i < page; i++) {
        if (flag[i] == false) return i;
    }
    return vt;
}

void Print(int total_page[10][100], int n, int page, int ref[100]) {
    // Print
    for (int i = 0; i < n; i++) {
        cout << ref[i] << " ";
    }
    cout << endl;
    for (int i = 0; i < page; i++) {
        for (int j = 0; j < n; j++) {
            if (total_page[i][j] != -1) {
                cout << total_page[i][j] << " ";
            }
            else {
                cout << " ";
            }
        }
        cout << endl;
    }
    for (int i = 0; i < n; i++) {
        if (page_fault[i] == 1) cout << "* ";
        else {
            cout << " ";
        }
    }
}

```

```

    }

    }

    cout << endl;
    cout << "Number of Page Fault: " << number_page_fault << endl;
}

void FIFO(int ref[], int n, int page)
{
    bool IsFault;
    int total_page[10][100];
    int current_page = 0;
    for (int i = 0; i < page; i++) {
        if (i == 0) { total_page[i][0] = ref[0]; }
        else {
            total_page[i][0] = -1;
        }
    }
    page_fault[0] = 1;

    for (int j = 1; j < n; j++) {
        for (int i = 0; i < page; i++) {
            total_page[i][j] = total_page[i][j-1];
            preArray[i] = total_page[i][j - 1];
        }
        if (Is_in_preArray(page, ref[j]) != -1) {
            page_fault[j] = -1;
        }
        else {
            current_page++;

```

```

        if (current_page == page) current_page = 0;
        total_page[current_page][j] = ref[j];
        page_fault[j] = 1;
        number_page_fault++;

    }

}

Print(total_page, n, page, ref);
}

void LRU(int ref[], int n, int page)
{
    bool IsFault;
    int total_page[10][100];
    int current_page = 0;
    for (int i = 0; i < page; i++) {
        if (i == 0) { total_page[i][0] = ref[0]; }
        else {
            total_page[i][0] = -1;
        }
    }
    page_fault[0] = 1;

    for (int j = 1; j < n; j++)
    {

        for (int i = 0; i < page; i++) {
            total_page[i][j] = total_page[i][j - 1];
            preArray[i] = total_page[i][j - 1];

```

```

    }
    /////
    if (Is_in_preArray(page, ref[j]) != -1) {
        page_fault[j] = 0;
    }
    else {
        if (j < page) {
            current_page++;
            total_page[current_page][j] = ref[j];
        }
        else {

            int pivot = Farest_Element(ref, j, page);
            total_page[pivot][j] = ref[j];
        }
        page_fault[j] = 1;
        number_page_fault++;
    }
    /////
}
Print(total_page, n, page, ref);
}

void OPT(int ref[], int n, int page)
{
    bool IsFault;
    int total_page[10][100];
    int current_page = 0;
    for (int i = 0; i < page; i++) {

```

```

        if (i == 0) { total_page[i][0] = ref[0]; }
        else {
            total_page[i][0] = -1;
        }
    }
    page_fault[0] = 1;

    for (int j = 1; j < n; j++)
    {

        for (int i = 0; i < page; i++) {
            total_page[i][j] = total_page[i][j - 1];
            preArray[i] = total_page[i][j - 1];
        }
        /////
        if (Is_in_preArray(page, ref[j]) != -1) {
            page_fault[j] = 0;
        }
        else {
            if (j < page) {
                current_page++;
                total_page[current_page][j] = ref[j];
            }
            else {

                int pivot = Fareast_Element_Oppsites(ref, j, page, n);
                total_page[pivot][j] = ref[j];
            }
            page_fault[j] = 1;
        }
    }

```



```

        number_page_fault++;

    }

    /////

}

Print(total_page, n, page, ref);
}

int main() {
    system("clear");
    int page; int temp;
    int ref[11] = { 2, 0, 5, 2, 1, 4, 4, 8, 0, 0, 7 };;
    int n = 11;
    cout << "--- Page Replacement algorithm ---" << endl;
    cout << "1. Default referenced sequence" << endl;
    cout << "2. Manual input sequence" << endl;
    cin >> temp;
    system("clear");
    switch (temp) {
        case 1:

            break;
        case 2:
            cout << "Nhap so luong: "; cin >> n;
            cout << "Nhap danh sach trang: ";
            for (int i = 0; i < n; i++) {
                cin >> ref[i];
            }
            system("clear");
    }
}

```

```
}
```

```
cout << "--- Page Replacement algorithm ---" << endl;  
cout << "Input page frames: "; cin >> page;  
system("clear");  
cout << "--- Select algorithm ---" << endl;  
cout << "1. FIFO algorithm" << endl;  
cout << "2. OPT algorithm" << endl;  
cout << "3. LRU algorithm" << endl;  
cout << "--- Enter input ---" << endl;
```

```
cin >> temp;  
system("clear");  
cout << "--- Page Replacement algorithm--- " << endl;  
switch (temp) {  
case 1:  
    FIFO(ref, n, page);  
    break;  
case 2:  
    OPT(ref, n, page);  
    break;  
case 3:  
    LRU(ref, n, page);  
    break;
```

```

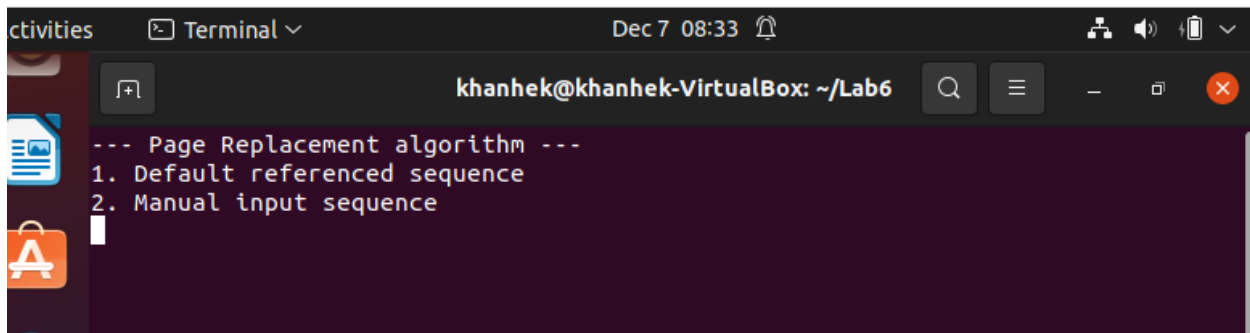
    }

    while(1);
    return 0;
}

```

Thực thi:

-Giao diện ban đầu:

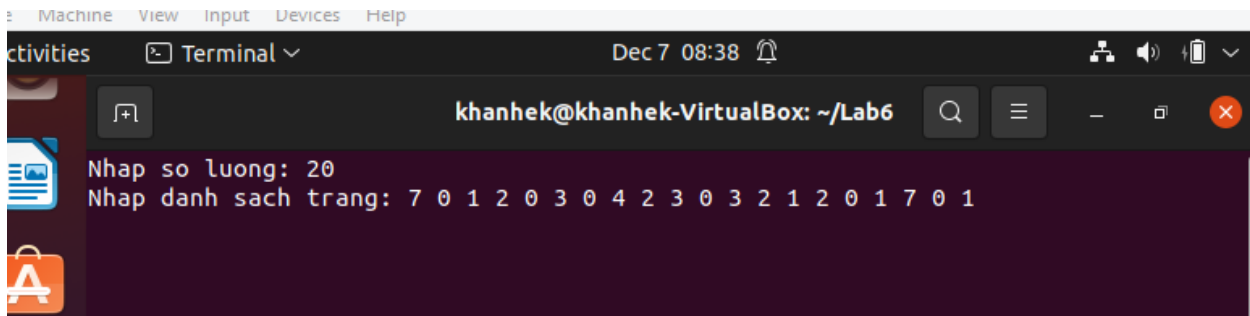


```

khanhek@khanhek-VirtualBox: ~/Lab6
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence

```

- Giao diện khi sau khi chọn Manual input sequence

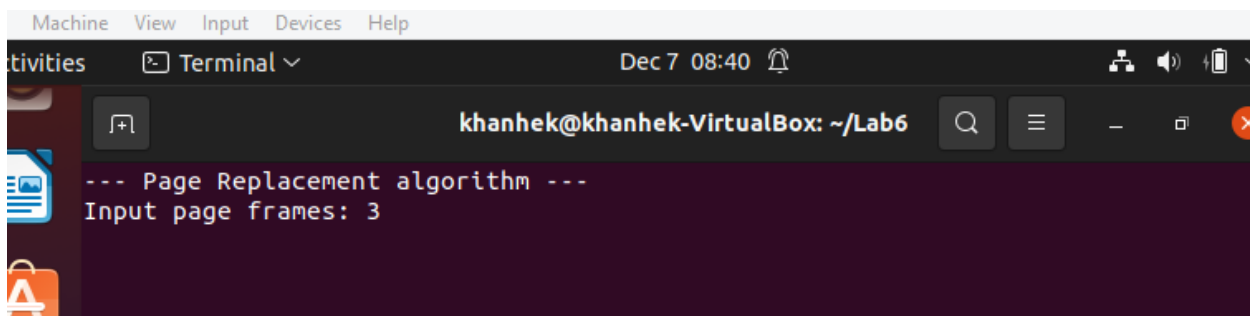


```

Machine View Input Devices Help
khanhek@khanhek-VirtualBox: ~/Lab6
Nhập số lượng: 20
Nhập danh sách trang: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

```

- Giao diện nhập Input page frames



```

Machine View Input Devices Help
khanhek@khanhek-VirtualBox: ~/Lab6
--- Page Replacement algorithm ---
Input page frames: 3

```

-Giao diện chọn giải thuật

```
activities Terminal Dec 7 08:41
khanhek@khanhek-VirtualBox: ~/Lab6
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
```

-Kết quả khi chọn giải thuật FIFO

```
khanhek@khanhek-VirtualBox: ~/Lab6
--- Page Replacement algorithm---
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
7 7 7 2 2 2 2 4 4 4 0 0 0 0 0 0 0 7 7 7
0 0 0 0 3 3 3 2 2 2 2 2 1 1 1 1 1 0 0
1 1 1 1 0 0 0 3 3 3 3 3 2 2 2 2 2 1
* * * * *
Number of Page Fault: 15
```

-Kết quả khi chọn giải thuật OPT

```
khanhek@khanhek-VirtualBox: ~/Lab6
--- Page Replacement algorithm---
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
7 7 7 2 2 2 2 2 2 2 2 2 2 2 2 2 2 7 7 7
0 0 0 0 0 0 0 4 4 4 0 0 0 0 0 0 0 0 0 0
1 1 1 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1
* * * * *
Number of Page Fault: 9
```

-Kết quả khi chọn giải thuật LRU

```
activities Terminal Dec 7 08:45
khanhek@khanhek-VirtualBox: ~/Lab6
--- Page Replacement algorithm---
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
7 7 7 2 2 2 2 4 4 4 0 0 0 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 3 3 3 3 3 3 0 0 0 0 0
1 1 1 3 3 3 2 2 2 2 2 2 2 2 2 2 7 7 7
* * * * *
Number of Page Fault: 12
```

- Kết quả khi chọn giải thuật FIFO (Nếu chọn Default referenced sequence)

```

File Machine View Input Devices Help
Activities Terminal Dec 7 08:48
khanhek@khanhek-Virtua
--- Page Replacement algorithm---
2 0 5 2 1 4 4 8 0 0 7
2 2 2 2 1 1 1 1 0 0 0
0 0 0 0 4 4 4 4 4 7
5 5 5 5 5 8 8 8 8
* * * * *
Number of Page Fault: 8

```

- Kết quả khi chọn giải thuật OPT (Nếu chọn Default referenced sequence)

```

khanhek@khanhek-VirtualBox: ~/Lab6
--- Page Replacement algorithm---
2 0 5 2 1 4 4 8 0 0 7
2 2 2 2 1 4 4 8 8 8 7
0 0 0 0 0 0 0 0 0 0 0
5 5 5 5 5 5 5 5 5 5
* * * * *
Number of Page Fault: 7

```

- Kết quả khi chọn giải thuật LRU (Nếu chọn Default referenced sequence)

```

khanhek@khanhek-Virtua
--- Page Replacement algorithm---
2 0 5 2 1 4 4 8 0 0 7
2 2 2 2 2 2 2 8 8 8 8
0 0 0 1 1 1 1 0 0 0
5 5 5 4 4 4 4 4 7
* * * * *
Number of Page Fault: 8

```

6.5/ Bài tập ôn tập

1. Nghịch lý Belady là gì? Sử dụng chương trình đã viết trên để chứng minh nghịch lý này.

- Thông thường, số frame tăng thì số page faults nên giảm. Tuy nhiên, với FIFO số frame tăng thì số page faults có thể cũng tăng theo. Điều nghịch lý này còn gọi là nghịch lý Belady.
- Ví dụ:

Khi có 3 frame page

```
--- Page Replacement algorithm---
1 2 3 4 1 2 5 1 2 3 4 5
1 1 1 4 4 4 5 5 5 5 5 5
  2 2 2 1 1 1 1 1 3 3 3
    3 3 3 2 2 2 2 2 4 4
* * * * * * * *
Number of Page Fault: 9
```

Khi có 4 frame page

```
--- Page Replacement algorithm---
1 2 3 4 1 2 5 1 2 3 4 5
1 1 1 1 1 1 5 5 5 5 4 4
  2 2 2 2 2 2 1 1 1 1 5
    3 3 3 3 3 3 2 2 2 2
      4 4 4 4 4 4 3 3 3
* * * * * * * *
Number of Page Fault: 10
```

2. Nhận xét về mức độ hiệu quả và tính khả thi của các giải thuật FIFO, OPT, LRU.

❖ Giải thuật nào là bất khả thi nhất? Vì sao?

Giải thuật OPT là bất khả thi nhất. Vì giải thuật sẽ thay thế trang lâu được sử dụng nhất trong tương lai

❖ Giải thuật nào là phức tạp nhất? Vì sao?

Giải thuật LRU là phức tạp nhất. Vì giải thuật cần sự hỗ trợ của phần cứng để lưu thông tin về thời gian page được tham chiếu