

Name: Phan Trọng Tính

MSSV: 21522683

Class: IT007-N17.1

## OPERATING SYSTEM

### LAB 6'S REPORT

#### SUMMARY

| Task |   | Status     | Page |
|------|---|------------|------|
| 6.4  |   | Hoàn thành | 2    |
| 6.5  | 1 | Hoàn thành | 19   |
|      | 2 | Hoàn thành | 20   |

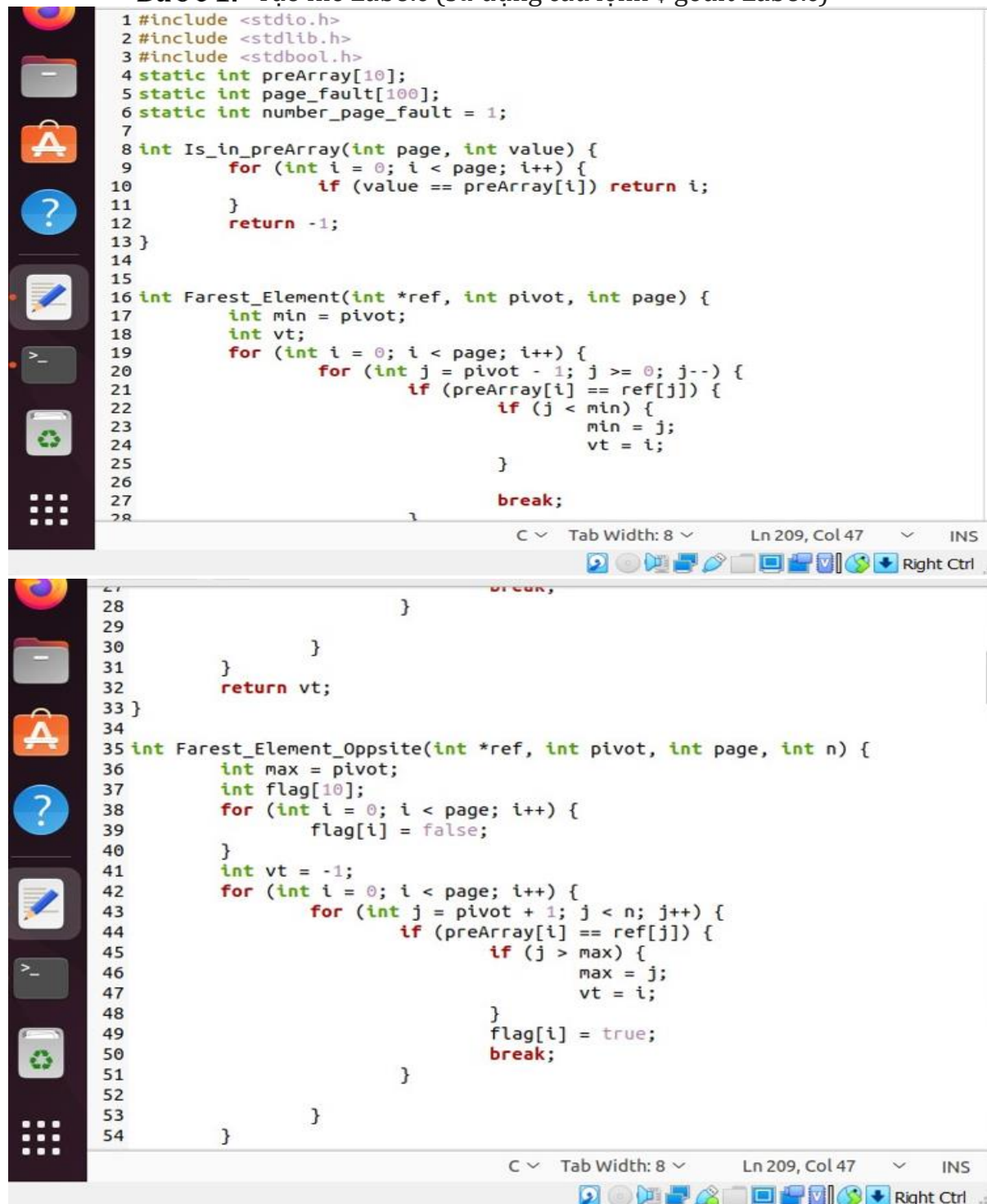
**Self-scores:**

*\*Note: Export file to **PDF** and name the file by following format:*

***LAB X – <Student ID>.pdf***

## 6.4 Hướng dẫn thực hành

- **Bước 1:** Tạo file Lab6.c (sử dụng câu lệnh \$ gedit Lab6.c)



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <stdbool.h>
4 static int preArray[10];
5 static int page_fault[100];
6 static int number_page_fault = 1;
7
8 int Is_in_preArray(int page, int value) {
9     for (int i = 0; i < page; i++) {
10         if (value == preArray[i]) return i;
11     }
12     return -1;
13 }
14
15
16 int Fareast_Element(int *ref, int pivot, int page) {
17     int min = pivot;
18     int vt;
19     for (int i = 0; i < page; i++) {
20         for (int j = pivot - 1; j >= 0; j--) {
21             if (preArray[i] == ref[j]) {
22                 if (j < min) {
23                     min = j;
24                     vt = i;
25                 }
26             }
27         }
28         break;
29     }
30 }
31
32 }
33 return vt;
34 }
35
36 int Fareast_Element_Oppsite(int *ref, int pivot, int page, int n) {
37     int max = pivot;
38     int flag[10];
39     for (int i = 0; i < page; i++) {
40         flag[i] = false;
41     }
42     int vt = -1;
43     for (int i = 0; i < page; i++) {
44         for (int j = pivot + 1; j < n; j++) {
45             if (preArray[i] == ref[j]) {
46                 if (j > max) {
47                     max = j;
48                     vt = i;
49                 }
50             }
51             flag[i] = true;
52             break;
53         }
54     }
55 }
```

```

55     for (int i = 0; i < page; i++) {
56         if (flag[i] == false) return i;
57     }
58     return vt;
59 }
60 }
61
62 void Print(int total_page[10][100], int n, int page, int ref[100]) {
63     // Print
64     for (int i = 0; i < n; i++) {
65         printf("%d ", ref[i]);
66     }
67     printf("\n");
68     for (int i = 0; i < page; i++) {
69         for (int j = 0; j < n; j++) {
70             if (total_page[i][j] != -1) {
71                 printf("%d ", total_page[i][j]);
72             }
73             else {
74                 printf(" ");
75             }
76         }
77         printf("\n");
78     }
79     for (int i = 0; i < n; i++) {
80         if (page_fault[i] == 1) printf("* ");
81         else {
82             printf(" ");

```

C Tab Width: 8 Ln 209, Col 47 INS

```

83     }
84 }
85 printf("\n");
86 printf("Number of Page Fault: %d\n", number_page_fault);
87 }
88
89 void FIFO(int ref[], int n, int page)
90 {
91     bool IsFault;
92     int total_page[10][100];
93     int current_page = 0;
94     for (int i = 0; i < page; i++) {
95         if (i == 0) { total_page[i][0] = ref[0]; }
96         else {
97             total_page[i][0] = -1;
98         }
99     }
100     page_fault[0] = 1;
101
102     for (int j = 1; j < n; j++) {
103         for (int i = 0; i < page; i++) {
104             total_page[i][j] = total_page[i][j-1];
105             preArray[i] = total_page[i][j - 1];
106         }
107         if (Is_in_preArray(page, ref[j]) != -1) {
108             page_fault[j] = -1;
109         }
110         else {

```

C Tab Width: 8 Ln 209, Col 47 INS

```

109         }
110         else {
111             current_page++;
112             if (current_page == page) current_page = 0;
113             total_page[current_page][j] = ref[j];
114             page_fault[j] = 1;
115             number_page_fault++;
116         }
117     }
118 }
119 Print(total_page, n, page, ref);
120 }
121
122 void OPT(int ref[], int n, int page)
123 {
124     bool IsFault;
125     int total_page[10][100];
126     int current_page = 0;
127     for (int i = 0; i < page; i++) {
128         if (i == 0) { total_page[i][0] = ref[0]; }
129         else {
130             total_page[i][0] = -1;
131         }
132     }
133     page_fault[0] = 1;
134
135     for (int j = 1; j < n; j++)
136

```

C Tab Width: 8 Ln 209, Col 47 INS

```

134
135     for (int j = 1; j < n; j++)
136     {
137
138         for (int i = 0; i < page; i++) {
139             total_page[i][j] = total_page[i][j - 1];
140             preArray[i] = total_page[i][j - 1];
141         }
142         /////
143         if (Is_in_preArray(page, ref[j]) != -1) {
144             page_fault[j] = 0;
145         }
146         else {
147             if (j < page) {
148                 current_page++;
149                 total_page[current_page][j] = ref[j];
150             }
151             else {
152
153                 int pivot = Fareast_Element(ref, j, page);
154                 total_page[pivot][j] = ref[j];
155             }
156             page_fault[j] = 1;
157             number_page_fault++;
158         }
159         ////
160     }
161     Print(total_page, n, page, ref);

```

C Tab Width: 8 Ln 209, Col 47 INS



```
160     }
161     Print(total_page, n, page, ref);
162 }
163
164 void LRU(int ref[], int n, int page)
165 {
166     bool IsFault;
167     int total_page[10][100];
168     int current_page = 0;
169     for (int i = 0; i < page; i++) {
170         if (i == 0) { total_page[i][0] = ref[0]; }
171         else {
172             total_page[i][0] = 0;
173         }
174     }
175     page_fault[0] = 1;
176
177     for (int j = 1; j < n; j++)
178     {
179
180         for (int i = 0; i < page; i++) {
181             total_page[i][j] = total_page[i][j - 1];
182             preArray[i] = total_page[i][j - 1];
183         }
184         /////
185         if (Is_in_preArray(page, ref[j]) != -1) {
186             page_fault[j] = 0;
187         }
188     }
189     else {
190         if (j < page) {
191             current_page++;
192             total_page[current_page][j] = ref[j];
193         }
194         else {
195             int pivot = Fareast_Element_Oppsite(ref, j,
196 page, n);
197             total_page[pivot][j] = ref[j];
198         }
199         page_fault[j] = 1;
200         number_page_fault++;
201     }
202     Print(total_page, n, page, ref);
203 }
204 }
205
206
207 int main() {
208     int page; int temp;
209     int ref[11] = { 2, 1, 5, 2, 1, 0, 8, 3, 0, 0, 7 };
210     int n = 11;
211     printf("--- Page Replacement algorithm ---\n");
212     printf("1. Default referenced sequence\n");
213     printf("2. Manual input sequence\n");
```

```

209     int ref[11] = { 2, 1, 5, 2, 1, 0, 8, 3, 0, 0, 7 };
210     int n = 11;
211     printf("--- Page Replacement algorithm ---\n");
212     printf("1. Default referenced sequence\n");
213     printf("2. Manual input sequence\n");
214     scanf("%d",&temp);
215     switch (temp) {
216     case 1:
217
218         break;
219     case 2:
220         printf("Nhap so luong: "); scanf("%d",&n);
221         printf("Nhap danh sach trang: ");
222         for (int i = 0; i < n; i++) {
223             scanf("%d",&ref[i]);
224         }
225     }
226     printf("--- Page Replacement algorithm ---\n");
227     printf("Input page frames: "); scanf("%d",&page);
228     printf("--- Select algorithm ---\n");
229     printf("1. FIFO algorithm\n");
230     printf("2. OPT algorithm\n");
231     printf("3. LRU algorithm\n");
232     printf("--- Enter input ---\n");
233
234     scanf("%d",&temp);
235     printf("--- Page Replacement algorithm--- \n");
236     switch (temp) {

```

C Tab Width: 8 Ln 209, Col 47 INS

```

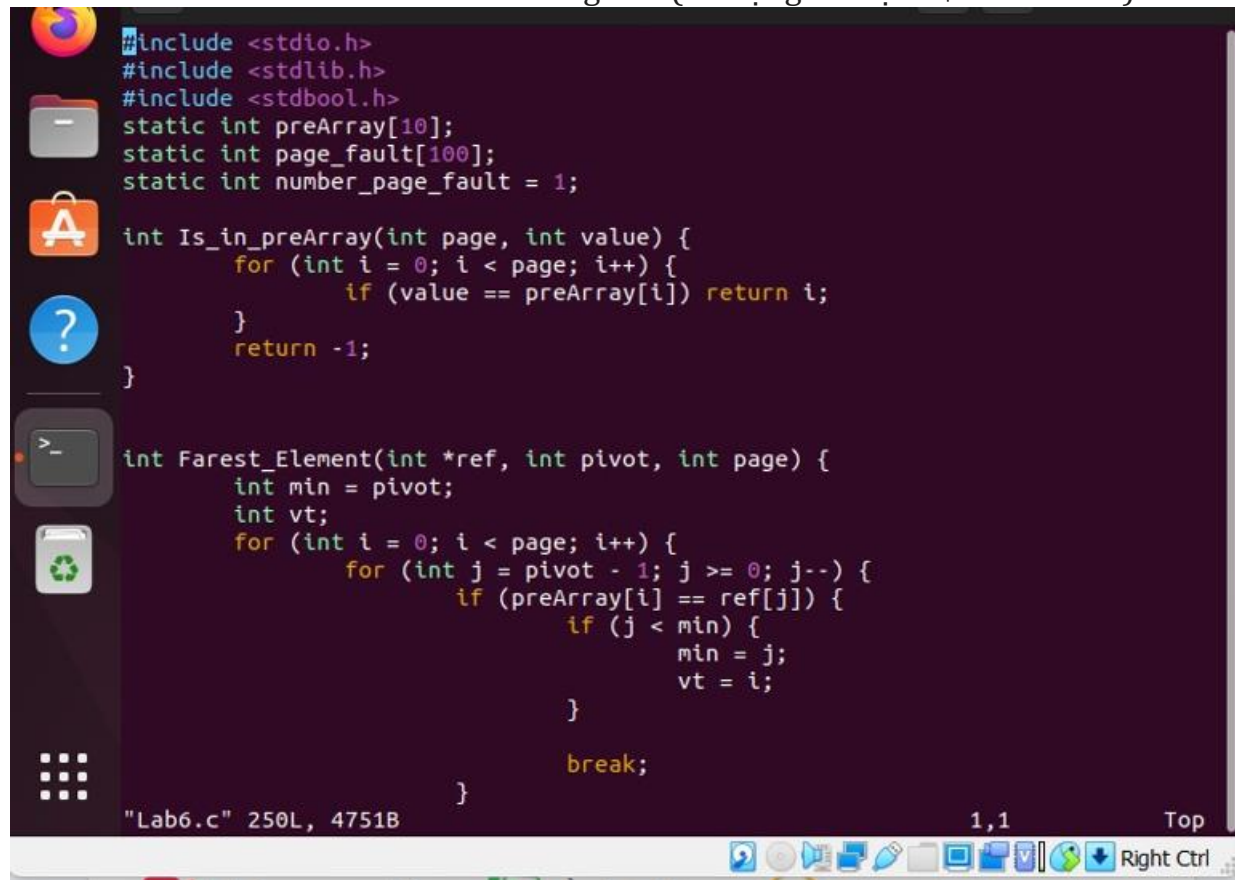
223         scanf("%d",&ref[i]);
224     }
225 }
226 printf("--- Page Replacement algorithm ---\n");
227 printf("Input page frames: "); scanf("%d",&page);
228 printf("--- Select algorithm ---\n");
229 printf("1. FIFO algorithm\n");
230 printf("2. OPT algorithm\n");
231 printf("3. LRU algorithm\n");
232 printf("--- Enter input ---\n");
233
234 scanf("%d",&temp);
235 printf("--- Page Replacement algorithm--- \n");
236 switch (temp) {
237 case 1:
238     FIFO(ref, n, page);
239     break;
240 case 2:
241     OPT(ref, n, page);
242     break;
243 case 3:
244     LRU(ref, n, page);
245     break;
246 }
247
248 system("pause");
249 return 0;
250 }

```

Lab6.c Save Right Ctrl

C Tab Width: 8 Ln 209, Col 47 INS

- **Bước 4:** Mở file Lab6.c bằng vim (sử dụng câu lệnh \$ vim Lab6.c)



```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
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 Fareast_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;
    }
}
```

"Lab6.c" 250L, 4751B 1,1 Top

- **Bước 5:** Thực thi file Lab6.c (sử dụng câu lệnh \$ gcc -pthread Lab6.c -o Lab6)

• **Bước 6:** Thực thi file Lab6 (sử dụng câu lệnh \$. / Lab6)

```

--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
1
--- Page Replacement algorithm ---
Input page frames: 3
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
1
--- Page Replacement algorithm---
2 1 5 2 1 0 8 3 0 0 7
2 2 2 2 2 0 0 0 0 0 7
1 1 1 1 1 8 8 8 8 8
5 5 5 5 5 3 3 3 3
* * * * *
Number of Page Fault: 7
sh: 1: pause: not found

--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
1
--- Page Replacement algorithm ---
Input page frames: 3
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
2
--- Page Replacement algorithm---
2 1 5 2 1 0 8 3 0 0 7
2 2 2 2 2 2 8 8 8 8 7
1 1 1 1 1 1 3 3 3 3
5 5 5 0 0 0 0 0 0
* * * * *
Number of Page Fault: 7
sh: 1: pause: not found

2 1 5 2 1 0 8 3 0 0 7
2 2 2 2 2 2 8 8 8 8 7
1 1 1 1 1 1 3 3 3 3
5 5 5 0 0 0 0 0 0
* * * * *
Number of Page Fault: 7
sh: 1: pause: not found

```



```
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
1
```

```
--- Page Replacement algorithm ---
Input page frames: 3
```

```
--- Select algorithm ---
```

```
Terminal algorithm
```

```
2. OPT algorithm
```

```
3. LRU algorithm
```

```
--- Enter input ---
```

```
3
```

```
--- Page Replacement algorithm---
```

```
2 1 5 2 1 0 8 3 0 0 7
```

```
2 2 2 2 0 0 0 0 0 7
```

```
0 1 1 1 1 1 8 3 3 3 3
```

```
0 0 5 5 5 5 5 5 5 5 5
```

```
* * * * *
```

```
Number of Page Fault: 7
```

```
sh: 1: pause: not found
```

```
--- Page Replacement algorithm ---
```

```
1. Default referenced sequence
```

```
2. Manual input sequence
```

```
2
```

```
Nhap so luong: 12
```

```
Nhap danh sach trang: 1 2 3 4 5 6 1 2 3 4 5 6
```

```
--- Page Replacement algorithm ---
```

```
Input page frames: 3
```

```
--- Select algorithm ---
```

```
1. FIFO algorithm
```

```
2. OPT algorithm
```

```
3. LRU algorithm
```

```
--- Enter input ---
```

```
1
```

```
--- Page Replacement algorithm---
```

```
1 2 3 4 5 6 1 2 3 4 5 6
```

```
1 1 1 4 4 4 1 1 1 4 4 4
```

```
2 2 2 5 5 5 2 2 2 5 5
```

```
3 3 3 6 6 6 3 3 3 6
```

```
* * * * *
```

```
Number of Page Fault: 12
```

```
--- Page Replacement algorithm ---
```

```
1. Default referenced sequence
```

```
2. Manual input sequence
```

```
2
```

```
Nhap so luong: 12
```

```
Nhap danh sach trang: 1 2 3 4 5 6 1 2 3 4 5 6
```

```
--- Page Replacement algorithm ---
```

```
Input page frames: 3
```

```
--- Select algorithm ---
```

```
1. FIFO algorithm
```

```
2. OPT algorithm
```

```
3. LRU algorithm
```

```
--- Enter input ---
```

```
Trash
```

```
--- Page Replacement algorithm---
```

```
1 2 3 4 5 6 1 2 3 4 5 6
```

```
1 1 1 4 4 4 1 1 1 4 4 4
```

```
2 2 2 5 5 5 2 2 2 5 5
```

```
3 3 3 6 6 6 3 3 3 6
```

```
* * * * *
```

```
Number of Page Fault: 12
```

```
sh: 1: pause: not found
```

```

--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhap so luong: 12
Nhap danh sach trang: 1 2 3 4 5 6 1 2 3 4 5 6
--- Page Replacement algorithm ---
Input page frames: 3
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
3
--- Page Replacement algorithm---
1 2 3 4 5 6 1 2 3 4 5 6
1 1 1 1 1 1 1 3 4 5 5
0 2 2 2 2 2 2 2 2 2 2
0 0 3 4 5 6 6 6 6 6 6
* * * * * * * *
Number of Page Fault: 9
sh: 1: pause: not found

```

## 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.

- + Số lượng lỗi trang xảy ra sẽ tăng lên khi số lượng khung trang sử dụng tăng. Hiện tượng này gọi là nghịch lý Belady.
- + Chứng minh:
  - Ví dụ: 1,2,3,4,1,2,5,1,2,3,4,5
  - Sử dụng giải thuật FIFO với 3 khung trang có 9 lỗi trang phát sinh

```
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhap so luong: 12
Nhap danh sach trang: 1 2 3 4 1 2 5 1 2 3 4 5
--- Page Replacement algorithm ---
Input page frames: 3
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
1
--- 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
sh: 1: pause: not found
```

- Sử dụng giải thuật FIFO với 4 khung trang có 10 lỗi trang phát sinh

```
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhap so luong: 12
Nhap danh sach trang: 1 2 3 4 1 2 5 1 2 3 4 5
--- Page Replacement algorithm ---
Input page frames: 4
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
1
--- 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
sh: 1: pause: not found
```

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

- ❖ Giải thuật nào bất khả thi nhất? Vì sao?
  - ❖ Giải thuật nào là phức tạp nhất? Vì sao?
- 
- + Giải thuật bất khả thi nhất là OPT, vì ta không thể đoán được trước là có những lỗi nào sẽ xảy ra để có thể đoán được vị trí lặp lại lâu nhất.
  - + Giải thuật phức tạp nhất là OPT, vì phải kiểm tra và cập nhật lại vị trí các lỗi trong tương lai sau mỗi lần kiểm tra lỗi, trong khi LRU chỉ cần cập nhật lại vị trí hiện tại nếu trùng lặp và thay thế nếu có vị trí nhỏ nhất. FIFO thì chỉ cần kiểm tra trùng lặp và đẩy ra khỏi hàng đợi