Name: Hồ Hồng Hà

ID: 20520480

Class:IT007.M13.1

OPERATING SYSTEM
LAB 6'S REPORT

SUMMARY

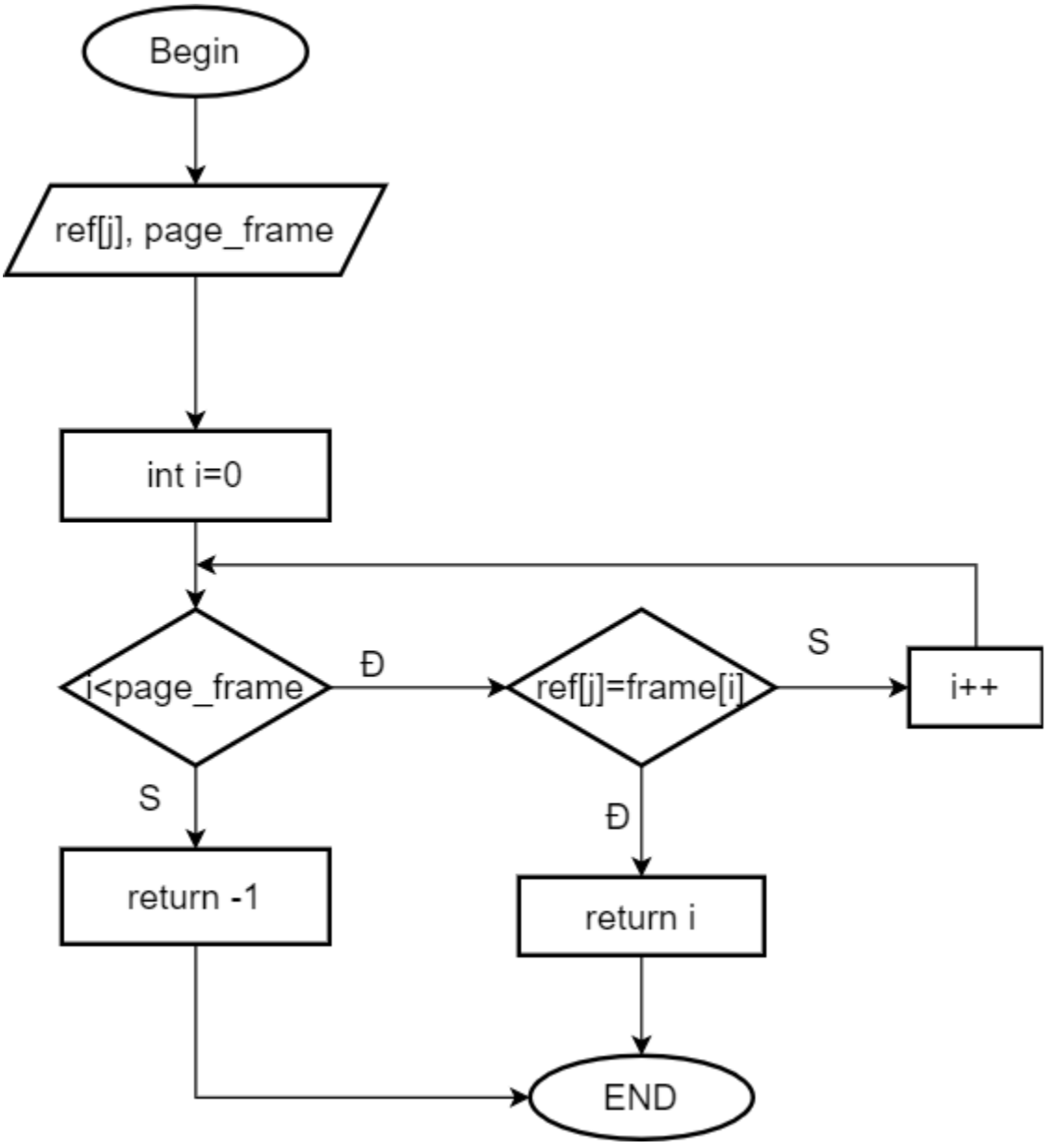
Self-scrores: 6

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

Section 6.4

1. Lưu đồ gải thuật và giải thích lưu đồ:

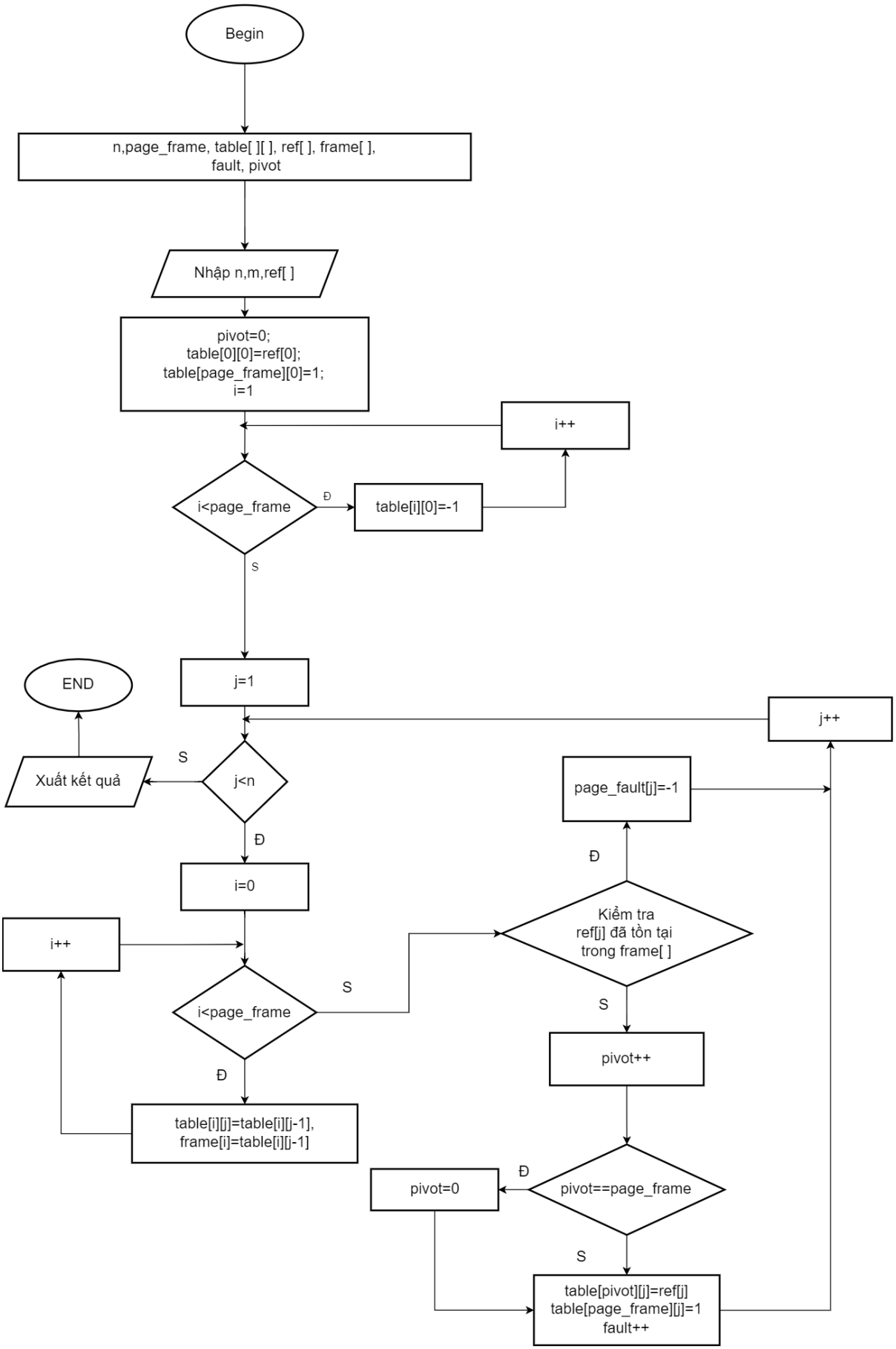
   *Lưu đồ kiểm tra giá trị truy xuất đã tồn tại trong khung trang chưa:



   Giải thích: Với mỗi gái trị ref[j] thuộc dãy truy xuất, duyệt qua lần lượt các giá trị trong mảng frame, nếu ref[j]=frame[j] có nghĩa là ref[j] đã tồn tại trong frame. Nếu không tồn tại thì trả về giá trị -1.

a. Giải thuật FIFO:

   *Lưu đồ gải thuật:


Begin
n,page_frame, table[ ][ ], ref[ ], frame[ ],
fault, pivot
Nhập n,m,ref[ ]
pivot=0;
table[0][0]=ref[0];
table[page_frame][0]=1;
i=1
i++
i<page_frame
Đ
table[i][0]=-1
S
j=1
j++
END
Xuất kết quả
S
j<n
Đ
i=0
i++
i<page_frame
S
Đ
table[i][j]=table[i][j-1],
frame[i]=table[i][j-1]
Kiểm tra
ref[j] đã tồn tại
trong frame[ ]
Đ
page_fault[j]=-1
S
pivot++
pivot==page_frame
Đ
pivot=0
S
table[pivot][j]=ref[j]
table[page_frame][j]=1
fault++

*Gải thích: Khởi tạo biến n để lưu số giá trị truy xuất, mảng ref[] lưu dãy truy xuất, mảng 2 chiều table[][] để lưu kết quả, page_frame để lưu số khung trang, fault=0 đến lưu số lỗi trang, pivot=0 là biến vị trí đưa trang truy xuất vào frame. Nhập các gái trị n,page_frame,ref[]. Đầu tiên gán table[0][0]=ref[0], table[page_frame][0]=1, fault++ để đưa giá trị truy xuất đầu tiên vào frame. Khởi tạo biến đếm i=1, trong khi i<page_frame thì gán table[i][0]=-1(gán bằng -1 là để khi xuất kết quả ra màn hình nếu là -1 thì bỏ qua). Gán j=1, trong khi j<n thì gán i=0:

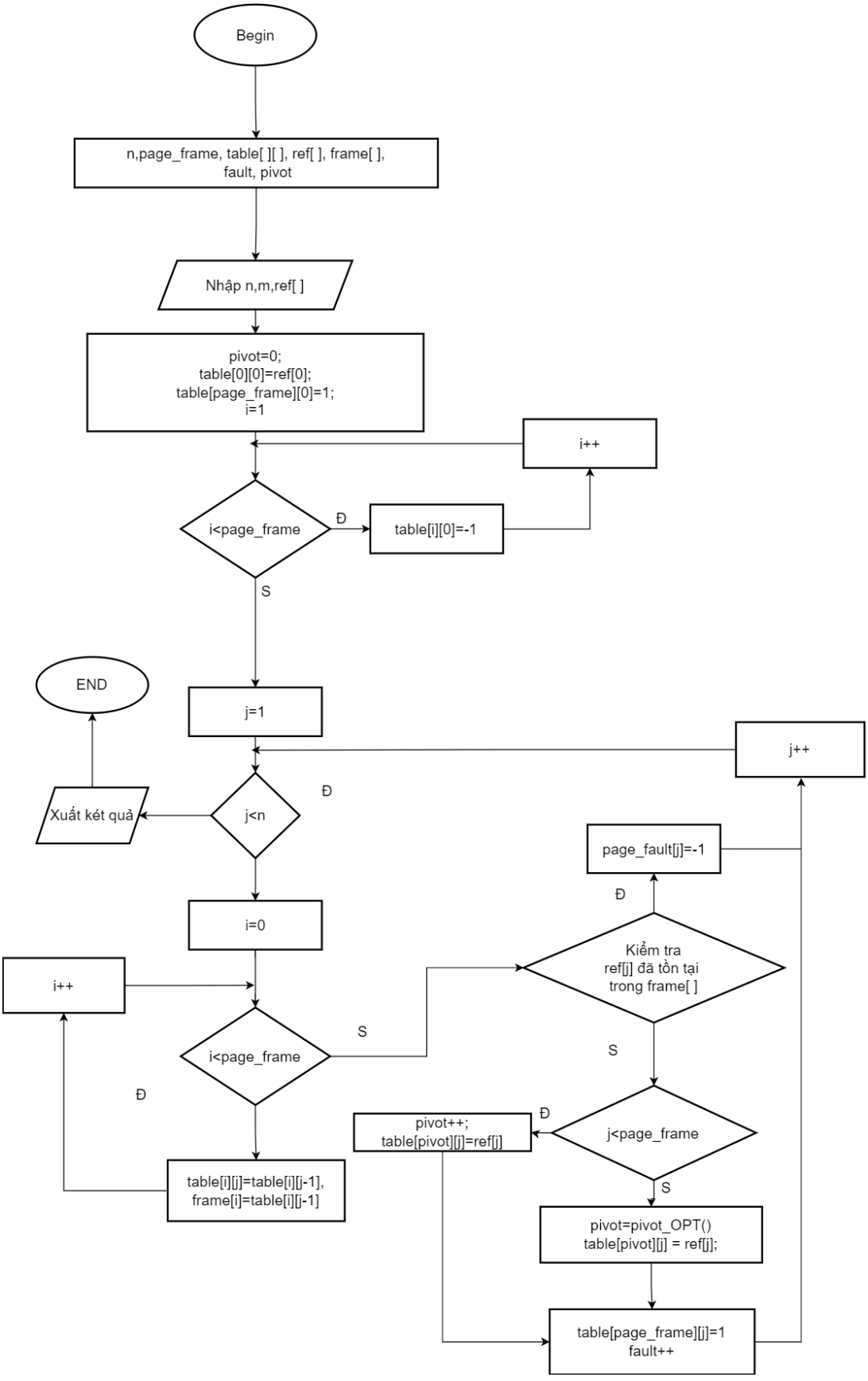
- Nếu i<page_frame thì gán table[i][j]=table[i][j-1] và frame[i]= table[i][j-1] sau đó tăng i lên 1 và tiếp tục so sánh I với page_frame rồi thực hiện tương tự.

-Nếu i>=page_frame thì kiểm tra ref[j] đã tồn tai trong frame hay chưa. Nếu đã tồn tại thì table[page_frame][j]=-1(nghĩa là không có lỗi) sau đó tăng j lên 1. Ngược lại thì tăng pivot lên 1 đơn vị sau đó so sánh pivot với page_frame nếu page_frame=pivot thì gán pivot=0. Sau khi so sánh xong thì gán table[pivot][j]=ref[j]; table[page_frame][j]=1; fault++

Tiếp tục tăng j lên thêm 1 và thực hiện cho đến khi j không bé hơn n thì kết thúc và xuất kết quả.

b. Giải thuật OPT

*Lưu đồ gải thuật OPT:


Begin
n,page_frame, table[ ][ ], ref[ ], frame[ ], fault, pivot
Nhập n,m,ref[ ]
pivot=0;
table[0][0]=ref[0];
table[page_frame][0]=1;
i=1
i++
i<page_frame
Đ
table[i][0]=-1
S
END
j=1
j++
Xuất két quả
j<n
Đ
page_fault[j]=-1
Đ
i=0
Kiểm tra ref[j] đã tồn tại trong frame[ ]
i++
i<page_frame
S
S
Đ
pivot++;
table[pivot][j]=ref[j]
Đ
j<page_frame
table[i][j]=table[i][j-1],
frame[i]=table[i][j-1]
S
pivot=pivot_OPT()
table[pivot][j] = ref[j];
table[page_frame][j]=1
fault++

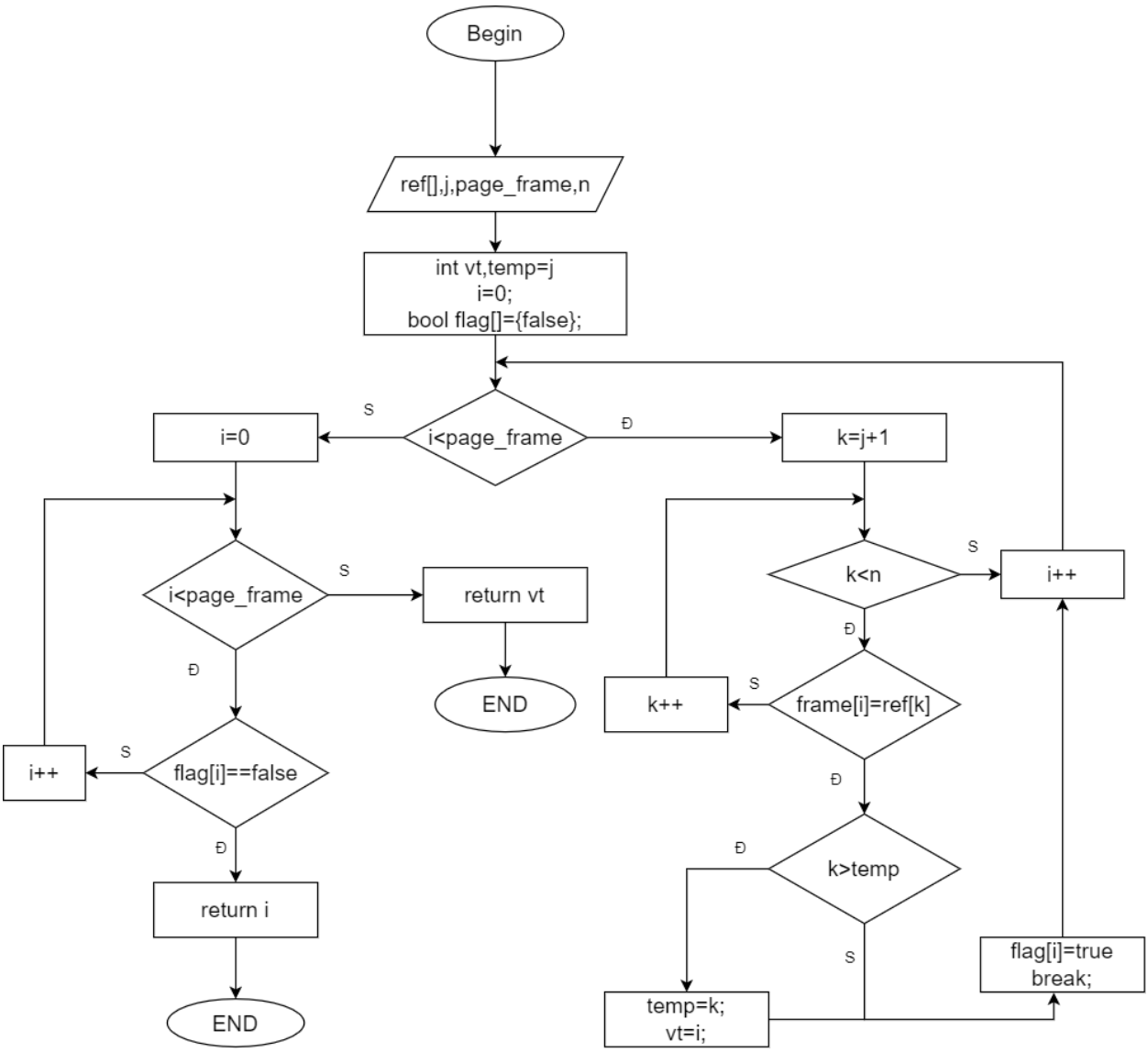
*Gải thích: Khởi tạo biến n để lưu số giá trị truy xuất, mảng ref[] lưu dãy truy xuất, mảng 2 chiều table[][] để lưu kết quả, page_frame để lưu số khung trang, fault=0 đến lưu số lỗi trang, pivot=0 là biến vị trí đưa trang truy xuất vào frame. Nhập các gái trị n,page_frame,ref[]. Đầu tiên gán table[0][0]=ref[0], table[page_frame][0]=1, fault++ để đưa giá trị truy xuất đầu tiên vào frame. Khởi tạo biến đếm i=1, trong khi i<page_frame thì gán table[i][0]=-1(gán bằng -1 là để khi xuất kết quả ra màn hình nếu là -1 thì bỏ qua). Gán j=1, trong khi j<n thì gán i=0:

- Nếu i<page_frame thì gán table[i][j]=table[i][j-1] và frame[i]= table[i][j-1] sau đó tăng i lên 1 và tiếp tục so sánh I với page_frame rồi thực hiện tương tự.

-Nếu i>=page_frame thì kiểm tra ref[j] đã tồn tai trong frame hay chưa. Nếu đã tồn tại thì table[page_frame][j]=-1(nghĩa là không có lỗi) sau đó tăng j lên 1 và quay trở lại so sánh j với n thực hiện tương tự. Ngược lại, kiểm tra nếu j<page_frame thì tăng pivot lên 1 đơn vị và gán table[pivot][j]=ref[j];]; table[page_frame][j]=1; fault++ còn nếu j>=page_frame thì tìm thực hiện tìm pivot sau đó gán table[pivot][j]=ref[j]

Tiếp tục tăng j lên thêm 1 và thực hiện cho đến khi j không bé hơn n thì kết thúc và xuất kết quả.

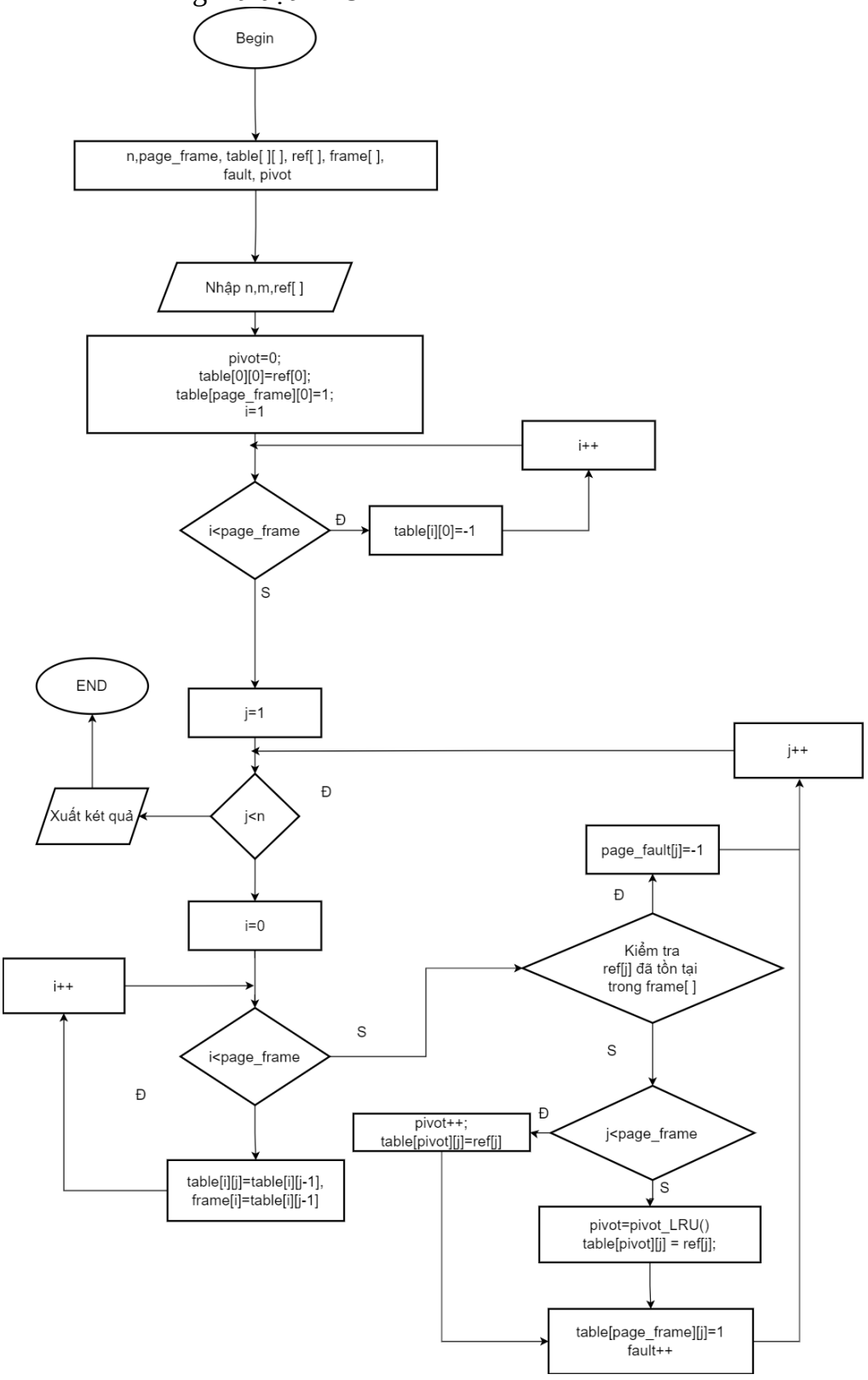
*Lưu đồ hàm tìm kiếm pivot_OPT(vị trí thêm vào trong gải thuật OPT)



*Giải thích: Hàm tìm kiếm sẽ có 4 tham số đầu vào là ref[], j, page_frame,n. Khởi tạo các biến vt(vt là vị trí khung trang để thêm gái trị ref[j] vào), temp=j,i=0,bool flag[]={false}. Mảng flag[] để đánh dấu rằng các gái trị trong mảng frame[] có còn sử dụng ở sau đó nữa không. Trong khi i<page_frame, khởi tạo k=j+1, trong khi k<n thì so sánh nếu frame[i]=ref[k] thì so sánh tiếp nếu k>temp (có nghĩa là giá trị ref[k] xuất hiện lại muộn hơn)thì gán temp=k,vt=i.Gán flag[i] = true và break quá trình lặp với biến k.Nếu k>=n thì i++ tiếp tục so sánh i với page_frame và thực hiện thương tự. Khi kết thúc vòng lặp gán i=0, so sánh nếu i<page_frame thì kiểm tra xem có vị trí nào trong flag[] có gái trị false (nghĩa là giá trị tại vị trí tương ứng trong frame không được sử dụng tiếp) trì trả về vị trí đó. Còn ngược lại trả về gái trị vt.

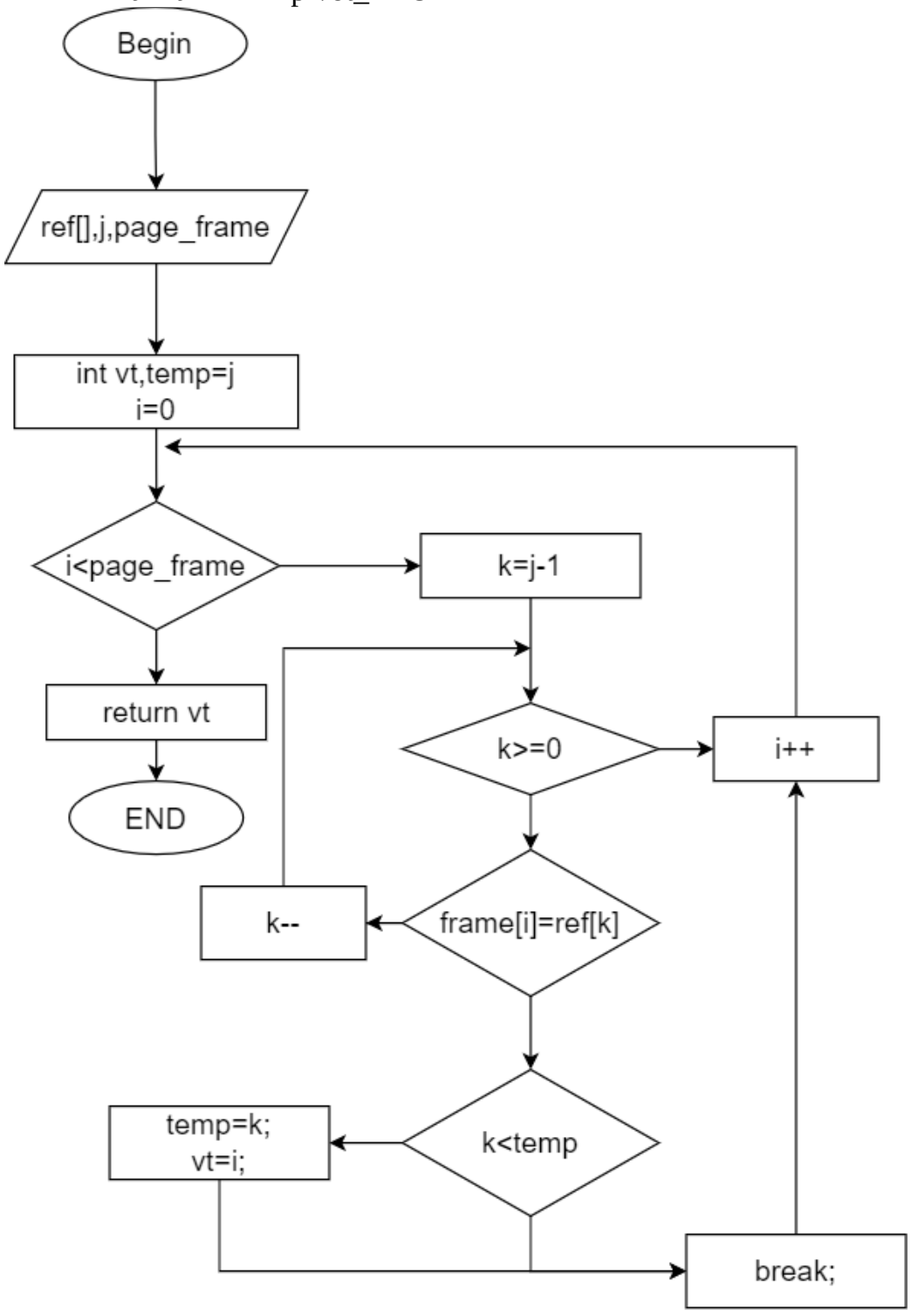
c. Giải thuật LRU

*Lưu đồ gải thuật LRU



*Giải thuật LRU thực hiện tương tự như giải thuật OPT chỉ thay đổi hàm tìm kiếm pivot_OPT thành pivot_LRU

*Lưu đồ hàm tìm kiếm pivot_LRU



*Giải thích:Hàm có 3 tham số đầu vào là ref[], j,page_frame. Khởi tạo các biến vt(vt là vị trí khung trang để thêm gái trị ref[j] vào), temp=j,i=0. Trong khi i<page_frame, khởi tạo k=j+1, trong khi k>=0 thì so sánh nếu frame[i]=ref[k] thì so sánh tiếp nếu k<temp (có nghĩa là giá trị ref[k]được sử dụng trước)thì gán temp=k,vt=i sau đó break vòng lặp với biến k.Nếu frame[i]!=ref[k] thì giảm k đi 1 và thực hiện tiếp cho đến khi k<0 thì tăng i lên thêm 1 và tiếp tục so sánh i với page_frame và tiếp tục thực hiện tương tự. Khi i không nhỏ hơn page_frame thì return vt và kết thúc.

2. Hiện thực bằng c++

```
#include <iostream>
using namespace std;
static int frame[20];
static int table[20][100];
static int fault = 0;
//Kiểm tra tồn tại trong frame
int Is_in_frame(int page_frame, int value) {
	for (int i = 0; i < page_frame; i++) {
		if (value == frame[i]) return i;
	}
	return -1;
}

//Tìm kiếm vị trí tahy thế trong giải thuật LRU
int pivot_LRU(int* ref, int pivot, int page_frame) {
	int min = pivot;
	int vt;
	for (int i = 0; i < page_frame; i++) {
		for (int j = pivot - 1; j >= 0; j--) {
			if (frame[i] == ref[j]) {
				if (j < min) {
					min = j;
					vt = i;
				}

				break;
			}

		}
	}
	return vt;
}
//Tìm kiếm vị trí tahy thế trong giải thuật OPT
int pivot_OPT(int* ref, int pivot, int page_frame, int n) {
	int max = pivot;
	int flag[20];
	for (int i = 0; i < page_frame; i++) {
		flag[i] = false;
	}
	int vt = -1;
	for (int i = 0; i < page_frame; i++) {
		for (int j = pivot + 1; j < n; j++) {
			if (frame[i] == ref[j]) {
				if (j > max) {
```

```
                                max = j;
                                vt = i;
                        }
                        flag[i] = true;
                        break;
                }

        }
    }

    for (int i = 0; i < page_frame; i++) {
        if (flag[i] == false) return i;
    }
    return vt;
}
//Xuất kết quả ra màn hình
void Print(int table[20][100], int n, int page_frame, int ref[100]) {
    for (int i = 0; i < n; i++) {
        cout << ref[i] << " ";
    }
    cout << endl;
    for (int i = 0; i < page_frame; i++) {
        for (int j = 0; j < n; j++) {
            if (table[i][j] != -1) {
                cout << table[i][j] << " ";
            }
            else {
                cout << "  ";
            }

        }
        cout << endl;
    }
    for (int i = 0; i < n; i++) {
        if (table[page_frame][i] == 1) cout << "* ";
        else {
            cout << "  ";
        }
    }
    cout << endl;
    cout << "Number of page_frame Fault: " << fault << endl;
}
//Giải thuật FIFO
void FIFO(int ref[], int n, int page_frame)
{

    int table[20][100];
    int pivot= 0;
    table[0][0] = ref[0]; table[page_frame][0] = 1;
    fault++;
    for (int i = 1; i < page_frame; i++) {
        table[i][0] = -1;
    }

    for (int j = 1; j < n; j++) {
        for (int i = 0; i < page_frame; i++) {
            table[i][j] = table[i][j - 1];
            frame[i] = table[i][j - 1];
```

```
			}
			if (Is_in_frame(page_frame, ref[j]) != -1) {
				table[page_frame][j] = -1;
			}
			else {
				pivot++;
				if (pivot== page_frame) pivot= 0;
				table[pivot][j] = ref[j];
				table[page_frame][j] = 1;
				fault++;

			}
		}
		Print(table, n, page_frame, ref);
}
//Giải thuật OPT
void OPT(int ref[], int n, int page_frame)
{

		int table[20][100];
		int pivot = 0;
		table[0][0] = ref[0]; table[page_frame][0] = 1;
		table[page_frame][0] = 1;
		fault++;
		for (int i = 1; i < page_frame; i++) {
			table[i][0] = -1;
		}

		for (int j = 1; j < n; j++)
		{
			for (int i = 0; i < page_frame; i++) {
				table[i][j] = table[i][j - 1];
				frame[i] = table[i][j - 1];
			}

			if (Is_in_frame(page_frame, ref[j]) != -1) {
				table[page_frame][j] = -1;
			}
			else {
				if (j < page_frame) {
					pivot++;
					table[pivot][j] = ref[j];
				}
				else {

					pivot = pivot_OPT(ref, j, page_frame, n);
					table[pivot][j] = ref[j];
				}
				table[page_frame][j] = 1;
				fault++;
			}
		}
		Print(table, n, page_frame, ref);
}
//Giải thuật LRU
void LRU(int ref[], int n, int page_frame)
{
```

```
	int table[20][100];
	int pivot = 0;
	table[0][0] = ref[0]; table[page_frame][0] = 1;
	table[page_frame][0] = 1;
	fault++;
	for (int i = 1; i < page_frame; i++) {
		table[i][0] = -1;
	}

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

		for (int i = 0; i < page_frame; i++) {
			table[i][j] = table[i][j - 1];
			frame[i] = table[i][j - 1];
		}
		if (Is_in_frame(page_frame, ref[j]) != -1) {
			table[page_frame][j] = -1;
		}
		else {
			if (j < page_frame) {
				pivot++;
				table[pivot][j] = ref[j];
			}
			else {

				pivot = pivot_LRU(ref, j, page_frame);
				table[pivot][j] = ref[j];
			}
			table[page_frame][j] = 1;
			fault++;
		}
	}
	Print(table, n, page_frame, ref);
}


int main() {
	int page_frame; int temp;
	int ref[50] = { 2, 0, 5, 2, 0, 4, 8, 0, 0, 0, 7 };;
	int n = 11;
	cout << "--- page_frame Replacement algorithm ---" << endl;
	cout << "1. Default referenced sequence" << endl;
	cout << "2. Manual input sequence" << endl;
	cin >> temp;
	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];
		}
	}

	cout << "--- page_frame Replacement algorithm ---" << endl;
```

```
        cout << "Input page_frame frames: "; cin >> page_frame;
        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;
        cout << "--- page_frame Replacement algorithm--- " << endl;
        switch (temp) {
        case 1:
               cout << "FIFO algorithm\n";
               FIFO(ref, n, page_frame);
               break;
        case 2:
               cout << "OPT algorithm\n";
               OPT(ref, n, page_frame);
               break;
        case 3:
               cout << "LRU algorithm\n";
               LRU(ref, n, page_frame);
               break;
        }
        return 0;
}
```

3. Kết quả thử nghiệm:

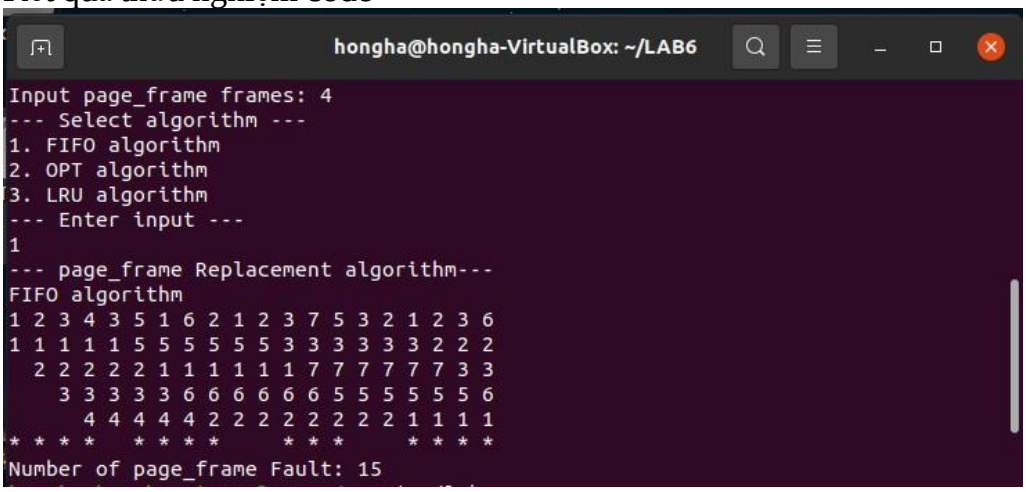
*Giải thuật FIFO:

Test case 1:

Kết quả chạy tay:

1	2	3	4	3	5	1	6	2	1	2	3	7	5	3	2	1	2	3	6
1	1	1	1	1	5	5	5	5	5	5	3	3	3	3	3	3	2	2	2
	2	2	2	2	2	1	1	1	1	1	1	7	7	7	7	7	7	3	3
		3	3	3	3	3	6	6	6	6	6	6	5	5	5	5	5	5	6
			4	4	4	4	4	2	2	2	2	2	2	2	2	1	1	1	1
*	*	*	*		*		*	*			*	*	*			*	*	*	*

Kết quả thửu nghiệm code



Test case 2:

Kết quả chạy tay:

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
*	*	*	*	*	*	*			*	*	

Kết quả chạy code

```
--- page_frame Replacement algorithm ---
Input page_frame frames: 3
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
1
--- page_frame Replacement algorithm---
FIFO 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_frame Fault: 9
```

Test case 3:

Kết quả chạy tay:

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
*	*	*	*		*	*	*	*	*	*			*	*			*	*	*

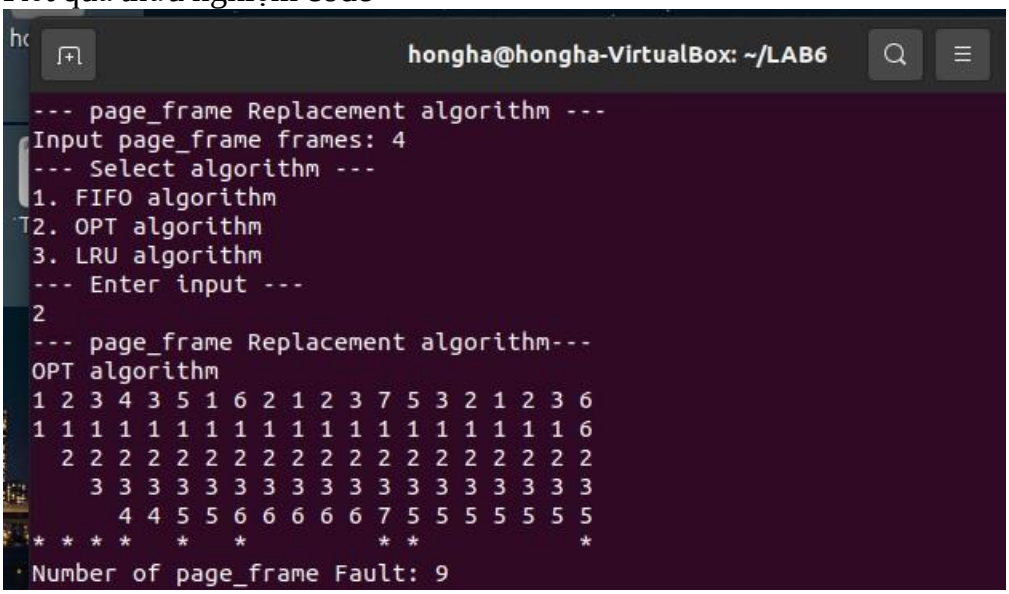
Kết quả chạy code:

```
--- page_frame Replacement algorithm ---
Input page_frame frames: 3
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
1
--- page_frame Replacement algorithm---
FIFO 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_frame Fault: 15
hongha@hongha-VirtualBox:~/LAB6$
```

Giải thuật OPT:
Test case 1:
Kết quả chạy tay:

1	2	3	4	3	5	1	6	2	1	2	3	7	5	3	2	1	2	3	6
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	6
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
			4	4	5	5	6	6	6	6	6	7	5	5	5	5	5	5	5
*	*	*	*		*		*					*	*						*

Kết quả thửu nghiệm code



Test case 2:
Kết quả chạy tay:

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	1	1	1	1	1	3	4	4
	2	2	2	2	2	2	2	2	2	2	2
		3	4	4	4	5	5	5	5	5	5
*	*	*	*			*			*	*	

Kết quả chạy code

```
--- page_frame Replacement algorithm ---
Input page_frame frames: 3
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
2
--- page_frame Replacement algorithm---
OPT algorithm
1 2 3 4 1 2 5 1 2 3 4 5
1 1 1 1 1 1 1 1 1 3 4 4
  2 2 2 2 2 2 2 2 2 2 2
    3 4 4 4 5 5 5 5 5 5
* * * *     *     * *
Number of page_frame Fault: 7
hongha@hongha-VirtualBox:~/LAB6$
```

Test case 3:

Kết quả chạy tay:

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	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
*	*	*	*		*		*			*			*				*		

Kết quả chạy code:

```
--- page_frame Replacement algorithm ---
Input page_frame frames: 3
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
2
--- page_frame Replacement algorithm---
OPT 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 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
* * * *   *   *     *     *       *
Number of page_frame Fault: 9
hongha@hongha-VirtualBox:~/LAB6$
```

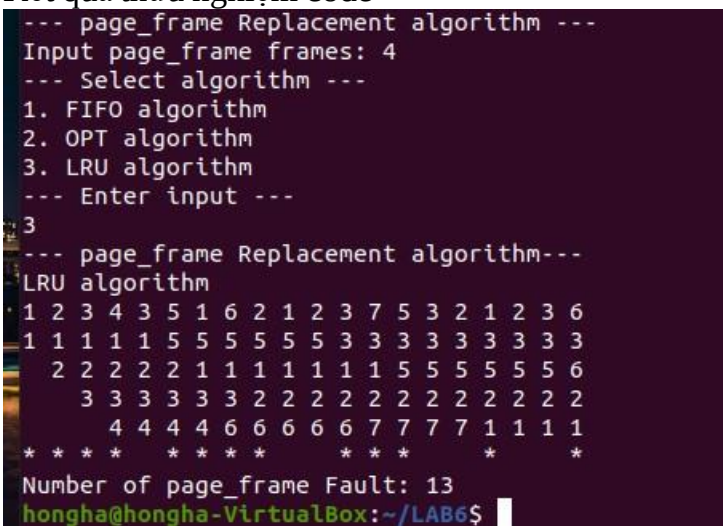
Giải thuật LRU:

Test case 1:

Kết quả chạy tay:

1	2	3	4	3	5	1	6	2	1	2	3	7	5	3	2	1	2	3	6
1	1	1	1	1	5	5	5	5	5	5	3	3	3	3	3	3	3	3	3
	2	2	2	2	2	1	1	1	1	1	1	1	5	5	5	5	5	5	6
		3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2
			4	4	4	4	6	6	6	6	6	7	7	7	7	1	1	1	1
*	*	*	*		*	*	*				*	*	*			*			*

Kết quả thửu nghiệm code



Test case 2:

Kết quả chạy tay:

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	4	4	4	5	5	5	3	3	3
	2	2	2	1	1	1	1	1	1	4	4
		3	3	3	2	2	2	2	2	2	5
*	*	*	*	*	*	*			*	*	*

Kết quả chạy code

```
--- page_frame Replacement algorithm ---
Input page_frame frames: 3
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
3
--- page_frame Replacement algorithm---
LRU algorithm
1 2 3 4 1 2 5 1 2 3 4 5
1 1 1 4 4 4 5 5 5 3 3 3
  2 2 2 1 1 1 1 1 1 4 4
    3 3 3 2 2 2 2 2 2 5
* * * * * * *     * * *
Number of page_frame Fault: 10
hongha@hongha-VirtualBox:~/LAB6$
```

Test case 3:

Kết quả chạy tay:

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	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	7	7	7
*	*	*	*		*		*	*	*	*			*		*		*		

Kết quả chạy code:

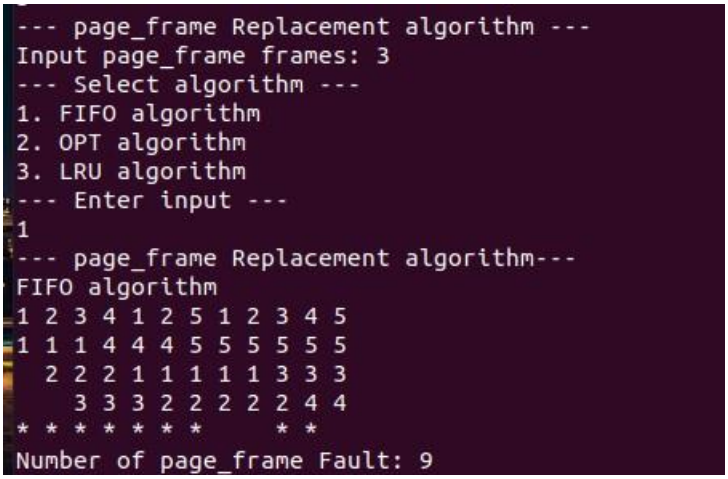
```
--- page_frame Replacement algorithm ---
Input page_frame frames: 3
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
3
--- page_frame Replacement algorithm---
LRU algorithm
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
7 7 7 2 2 2 2 4 4 4 0 0 0 1 1 1 1 1 1 1
  0 0 0 0 0 0 0 0 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 7 7 7
* * * *   *   * * * *     *   *   *
Number of page_frame Fault: 12
hongha@hongha-VirtualBox:~/LAB6$
```

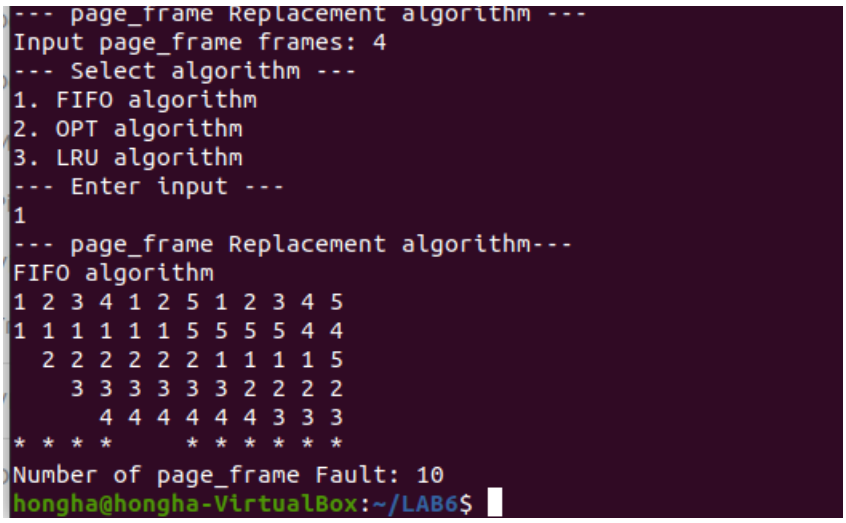
Section 6.5

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

   Nghịch lý Belady là hiện tượng số lượng lỗi trang xảy ra sẽ tăng lên khi số lượng khung trang sử dụng tăng.

   Ví dụ với cùng giải thuật FIFO và cùng 1 test case: khi số khung trang là 3 thì số lỗi là 9 còn khi số khung trang là 4 thì số lỗi là 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 nào là phức tạp nhất? Vì sao?

   FIFO: Hiệu quả không cao, dễ cài đặt, tính khả thi cao

   LRU: Hiệu quả tốt, khó cài đặt

   OPT: Hiệu quả cao nhưng không khả thi, khó cài đặt

   Giải thuật bất khả thi nhất là OPT vì không biết các trang nào sẽ xuất hiện như thế nào trong tương lai.

   Cài đặt OPT và LRU phức tạp nhất vì phải xét đến toàn bộ dãy trước đó đối với giải thuật LRU và dãy phía sau đối với giải thuật OPT