Name: NGUYỄN ANH TÀI - ID: 20520924

Class: IT007.M22.1

OPERATING SYSTEM
LAB 05'S REPORT

SUMMARY

Task	Status
1. Thực hiện bài tập trong phần 6.4	
giải thuật FIFO	DONE
giải thuật OPT	
giải thuật LRU	
2. Bài tập phần 6.5 là phần bonus (2 điểm)	DONE

Self-scrores: 5

Task 1:

a. Giải thuật thay thế trang FIFO.

```cpp
#include <iostream>
#include <string>
#include <stdio.h>
#include <stdlib.h>
#include <vector>
#include <algorithm>
#include <sstream>
using namespace std;
int PageFrame = 0;
int Pages = 0;
int numberOfPagesFault = 0;
struct PAGE
{
    int Index;
    int NUMBEROUS = 0;
    int PAST_REFERENCE_TIME = 0;
    int FURTURE_REFERENCE_TIME = 0;
    int FAULT_PAGE = 0;
    int *FRAME_TABLE = new int[PageFrame];
    // int FRAME_TABLE[3] = {0, 0, 0};
};

bool isExist(int currentFrame[], int value)
{

    for (int j = 0; j < PageFrame; j++)
    {
        if (currentFrame[j] == value)
        {
            return true;
        }
    }
    return false;
}

void FIFO_Replacer(PAGE *&PageTable)
{
    int currentFrame[Pages][1000];
    // Khoi tao cac gia tri maac dinh
    for (int i = 0; i < Pages; i++)
```

```
{
    for (int j = 0; j < PageFrame; j++)
        currentFrame[i][j] = -1;
}

for (int i = 0; i < PageFrame; i++)
{

    for (int j = i; j < PageFrame; j++)
        currentFrame[j][i] = PageTable[i].Index;
}
for (int i = 0; i < PageFrame; i++)
{
    PageTable[i].FAULT_PAGE = 1;
    PageTable[i].FRAME_TABLE = currentFrame[i];
}

int Smallest = 0;
for (int i = PageFrame; i < Pages; i++)
{
    if (!isExist(currentFrame[i - 1], PageTable[i].Index))
    {
        for (int j = 0; j < PageFrame; j++)
        {
            currentFrame[i][j] = currentFrame[i - 1][j];
        }
        currentFrame[i][Smallest] = PageTable[i].Index;
        PageTable[i].FRAME_TABLE = currentFrame[i];

        PageTable[i].FAULT_PAGE = 1;
        Smallest++;
    }
    else
    {
        PageTable[i].FRAME_TABLE = currentFrame[i - 1];
        PageTable[i].FAULT_PAGE = 0;
        for (int j = 0; j < PageFrame; j++)
        {
            currentFrame[i][j] = currentFrame[i - 1][j];
        }
    }
    if (Smallest == PageFrame)
    {
        Smallest = 0;
    }
```

```
    }
}

string space(int n)
{
    string res;
    for (int i = 0; i < n; i++)
    {
        res += " ";
    }
    return res;
}

vector<string> AddFaultPage(int num_columns, PAGE *PageTable, string *Attribute)
{
    vector<string> row;
    string *Value = new string[Pages + 1];
    for (int i = 0; i < Pages + 1; i++)
    {
        if (i == 0)
            Value[0] = " ";
        else if (PageTable[i - 1].FAULT_PAGE != 0)
        {
            Value[i] = "*";
            numberOfPagesFault++;
        }
    }

    for (int j = 0; j < num_columns; j++)
    {
        stringstream ss;
        int num;
        ss << Value[j];
        ss >> num;
        if (j == 0)
            row.push_back("|" + space(Attribute[0].length() - Value[0].length()) +
Value[0] + "|");
        else if (num >= 0)
            row.push_back(space(Attribute[j].length() - Value[j].length()) + Value[j] +
"|");
        else
            row.push_back(space(Attribute[j].length() - Value[j].length()) + " " + "|");
    }
    return row;
}
```

```
void Manual_Input_Sequence(PAGE *&PageTable)
{

    cout << "Input page frames: ";
    cin >> PageFrame;
    cout << "Input the number of Page: ";
    cin >> Pages;

    PAGE *newArr = new PAGE[Pages];

    for (int i = 0; i < Pages; i++)
    {
        cin >> newArr[i].Index;
        newArr[i].NUMBEROUS = i;
    }
    PageTable = newArr;
}

void Output(int Pages, int PageFrame, PAGE *PageTable, string NameOfAlthorithm)
{

    string *Attribute = new string[Pages + 1];
    for (int i = 0; i < Pages + 1; i++)
    {
        if (i == 0)
            Attribute[0] = "Page Frame";
        else
            Attribute[i] = to_string(PageTable[i - 1].Index);
    }
    int length = Attribute[0].length();

    int num_columns = Pages + 1;
    vector<vector<string>> table;

    vector<string> row{"\t" + NameOfAlthorithm};
    table.push_back(row);

    for (int i = 0; i < PageFrame + 1; i++)
    {
        vector<string> row;

        if (i == 0)
        {
            // First Row
```

```
            for (int j = 0; j < num_columns; j++)
            {
                if (j == 0)
                    row.push_back("|" + Attribute[0] + "|");
                else
                    row.push_back(Attribute[j] + "|");
            }
            table.push_back(row);
        }
        else
        {
            // Another Row
            int ele = i - 1;
            string *Value = new string[Pages + 1];
            for (int i = 0; i < Pages + 1; i++)
            {
                if (i == 0)
                    Value[0] = " ";
                else
                    Value[i] = to_string(PageTable[i - 1].FRAME_TABLE[ele]);
            }

            for (int j = 0; j < num_columns; j++)
            {
                stringstream ss;
                int num;
                ss << Value[j];
                ss >> num;
                if (j == 0)
                    row.push_back("|" + space(Attribute[0].length() - Value[0].length())
+ Value[0] + "|");
                else if (num >= 0)
                    row.push_back(space(Attribute[j].length() - Value[j].length()) +
Value[j] + "|");
                else
                    row.push_back(space(Attribute[j].length() - Value[j].length()) + " "
+ "|");
            }
            table.push_back(row);
        }
    }
    row = AddFaultPage(num_columns, PageTable, Attribute);
    table.push_back(row);
    row = {"Number of FAULT_PAGE:" + to_string(numberOfPagesFault)};
    table.push_back(row);
```

```
    // Print The Table
    for (int i = 0; i < PageFrame + 4; i++) // number of rows
    {
        for (int j = 0; j < num_columns; j++)
        {
            if (i == Pages + 2)
            {
                cout << table[i][0].c_str();
                cout << table[i][1].c_str();
                break;
            }
            else if (i == 0)
            {
                cout << table[i][0].c_str();
                break;
            }
            else if (i == PageFrame + 3 && j == 0)
            {
                cout << table[i][j].c_str();
                break;
            }
            cout << table[i][j].c_str();
        }
        cout << "\n";
    }
}

int main()
{

    Pages = 1;
    PAGE *PageTable = new PAGE[Pages];
    Manual_Input_Sequence(PageTable);
    FIFO_Replacer(PageTable);
    Output(Pages, PageFrame, PageTable, "FIFO");
    system("pause");
    return 0;
}
```

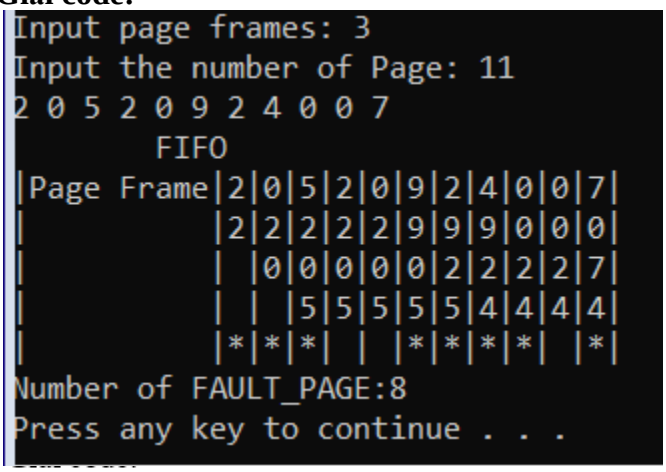
Tính đúng đắn của giải thuật FIFO:

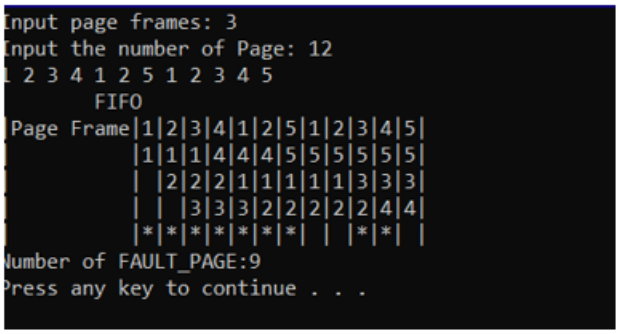
Giải tay:

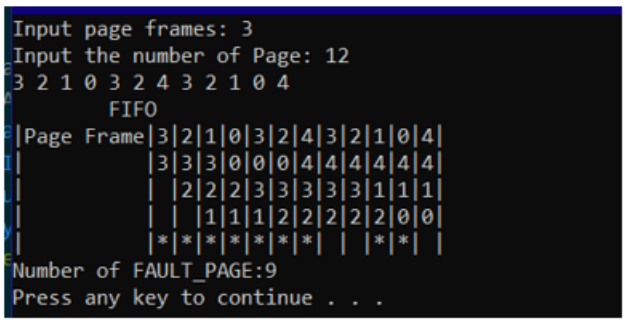
2	0	5	2	0	9	2	4	0	0	7
2	2	2	2	2	9	9	9	0	0	0
	0	0	0	0	0	2	2	2	2	7
		5	5	5	5	5	4	4	4	4
*	*	*			*	*	*	*		*

Có 8 lỗi trang

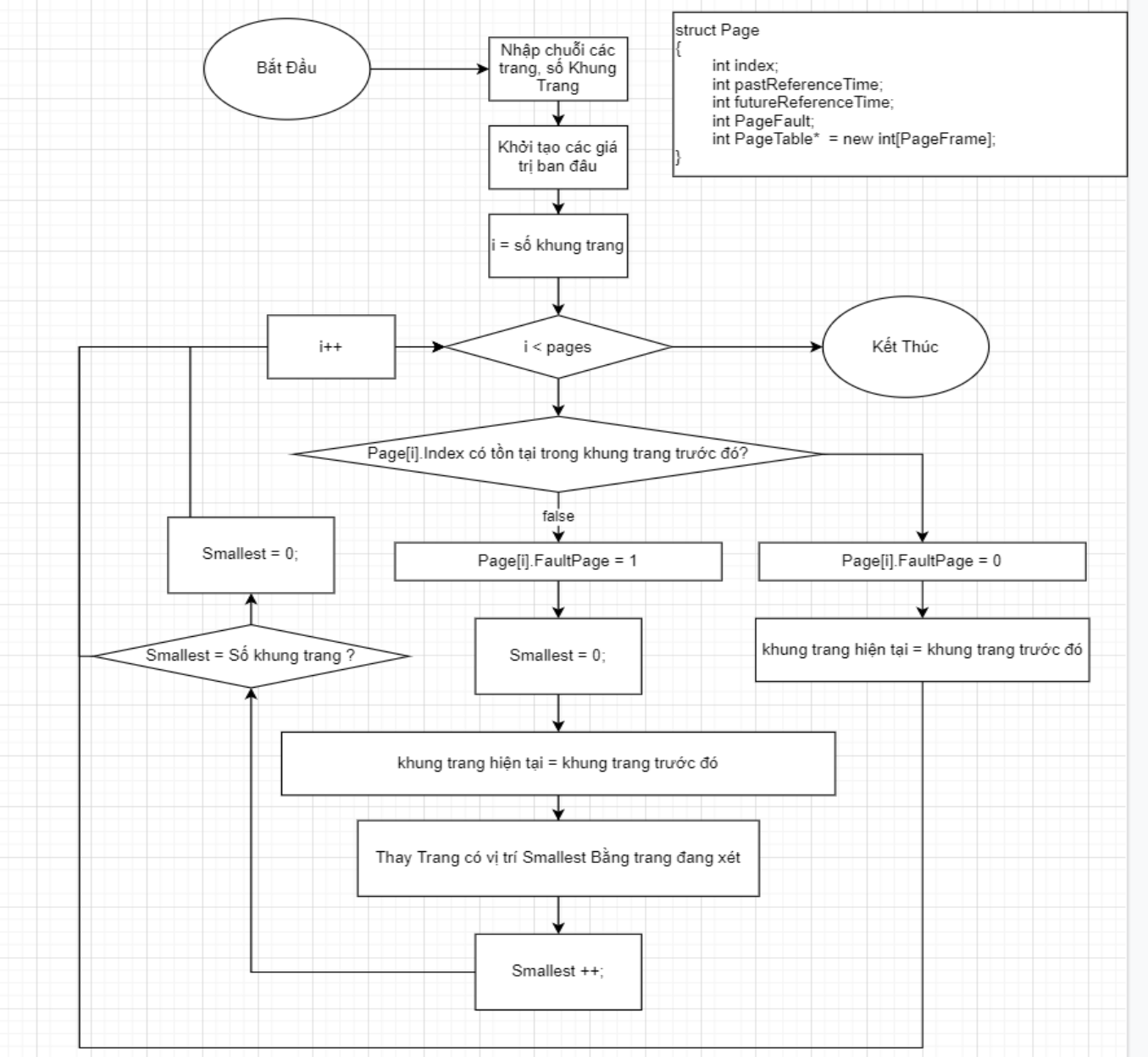
Giải code:







Lưu đồ giải thuật:



Diễn Giải:

- Sau khi thông số cần thiết cho bài toán ta sẽ tạo các khung trang "mồi" ở các khung trang đầu tiền khi chưa bị đầy
- bắt đầu từ i = số khung trang ta sẽ kiểm tra lần lượt các trang đang xét có tồn tại trong khung trang trước đó hay không

+nếu không ta sẽ tiến hành thay thế khung trang dựa vào lưu đồ giải thuật trên theo các bước sau:

1. Khung trang hiện tại = khung trang trước đó
2. Thay thế trang có vị trí Smallest bằng Trang đang xét.
3. Tăng dần và đảm bảo cho Smallest luôn nằm trong [0; Số trang-1]

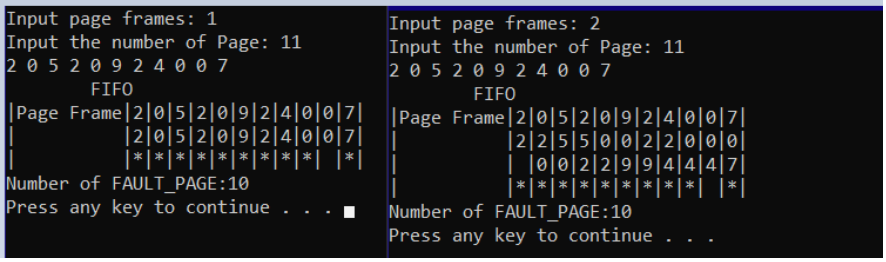
+Nếu có thì ta sẽ gán khung trang hiện tại bằng khung trang trước đó và gán PageFault = 0

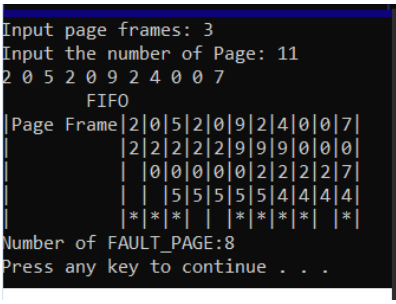
Task 2:

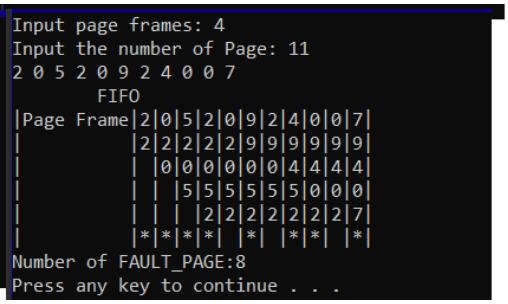
1. Nghịch lý Belady là gì? Sử dụng chương trình đã viết trên để chứng minh nghịch lý này.

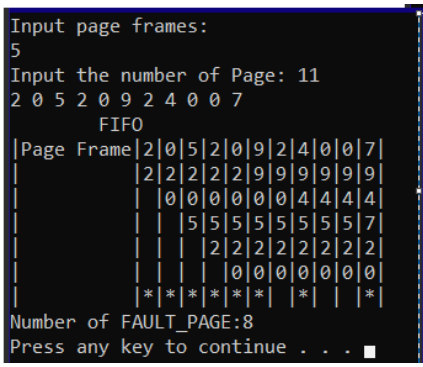
Nghịch lý Belady là hiện tượng tăng số lượng khung trang dẫn đến tăng số lỗi trang đối với các kiểu truy cập bộ nhớ nhất định. Hiện tượng này thường gặp phải khi sử dụng thuật toán thay thế trang FIFO. Trong FIFO, lỗi trang có thể tăng hoặc không thể tăng khi số khung trang tăng lên, nhưng trong các thuật toán tối ưu và dựa trên ngăn xếp như LRU, khi số lượng khung trang tăng lên, lỗi trang sẽ giảm.

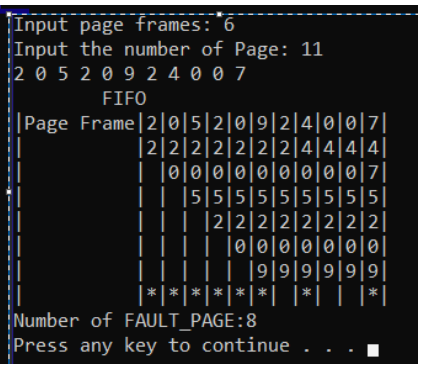
Ta có các testcase thể hiện sự tăng giảm khung trang tương ứng với kết quả từ giải thuật FIFO:







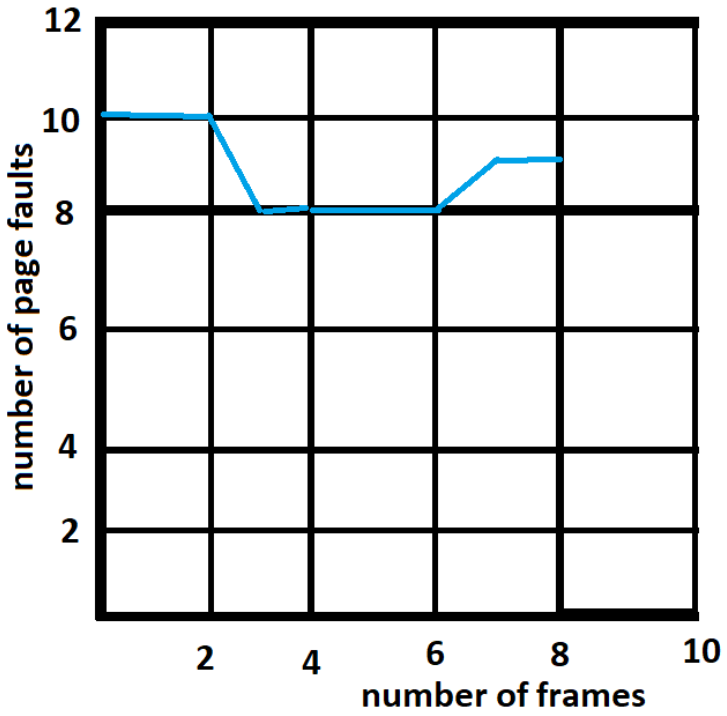




```
Input page frames: 7
Input the number of Page: 11
2 0 5 2 0 9 2 4 0 0 7
        FIFO
|Page Frame|2|0|5|2|0|9|2|4|0|0|7|
|          |2|2|2|2|2|2|2|4|4|4|4|
|          | |0|0|0|0|0|0|0|0|0|7|
|          | | |5|5|5|5|5|5|5|5|5|
|          | | | |2|2|2|2|2|2|2|2|
|          | | | | |0|0|0|0|0|0|0|
|          | | | | | |9|9|9|9|9|9|
|          | | | | | | |2|2|2|2|2|
|          |*|*|*|*|*|*|*| | |*|
Number of FAULT_PAGE:9
Press any key to continue . . .
```

```
Input page frames: 8
Input the number of Page: 11
2 0 5 2 0 9 2 4 0 0 7
        FIFO
|Page Frame|2|0|5|2|0|9|2|4|0|0|7|
|          |2|2|2|2|2|2|2|2|2|2|7|
|          | |0|0|0|0|0|0|0|0|0|0|
|          | | |5|5|5|5|5|5|5|5|5|
|          | | | |2|2|2|2|2|2|2|2|
|          | | | | |0|0|0|0|0|0|0|
|          | | | | | |9|9|9|9|9|9|
|          | | | | | | |2|2|2|2|2|
|          | | | | | | | |4|4|4|4|
|          |*|*|*|*|*|*|*|*| | |*|
Number of FAULT_PAGE:9
Press any key to continue . . .
```

Từ những test case trên ta có được một biểu đồ như sau



- Như vậy: Ta có thể thấy được sự tác động của việc thay đổi số khung trang ảnh hưởng tới lỗi trang cụ thể là khi tăng số trung trang thì sô lỗi trang có xu hướng giảm xuống.

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

- Nhận xét:
  + Giải thuật FIFO: dễ dàng cài đặt, dễ hiện thực, kém hiệu quả
  + Giải thuật LRU: khó cài đặt, phức tạp, hiệu quả.
  + Giải thuật OPT: không khả thi, hiệu quả nhất
- Trong các giải thuật không có tính khả thi là OPT vì việc biết trước những trang nào có thể được truy xuất tiếp theo là điều không thể.

Giải thuật phức tạp nhất là OPT và LRU vì mỗi lần lỗi trang, khi tìm khung trang thích hợp để thay thế rất khó khăn.