ใบงานที่ 9

เรื่อง Page Replacement

เสนอ

อาจารย์ปิยพล ยืนยงสถาวร

จัดทำโดย

นาย กวีวัธน์ กาญจนสุพัฒนากุล 65543206003-7

ใบงานนี้เป็นส่วนหนึ่งของรายวิชา ระบบปฏิบัติการ

หลักสูตรวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา

ประจำภาคที่ 2 ปีการศึกษา 2566

Lab 9 : Page Replacement

ลำดับขั้นการทดลอง

1. จงแสดงวิธีการแทนที่หน่วยความจำ และออกแบบโปรแกรมจำลองการแทนที่หน่วยความจำ
2. จงเขียนโปรแกรมจำลองการทำงานของ Algorithm ของ Page Replacement

Consider the following page-reference string for one process:

2, 1, 2, 3, 7, 6, 2, 3, 4, 2, 1, 5, 6, 3, 2, 3, 6, 2, 1.

How many page faults would occur for the below replacement algorithms,

assuming a total 4 frames available for the process.

Assume pure demand paging (all frames are initially empty),

so the first page faults will all cost one fault each.

☐ Least Recently Used (FIFO) replacement

2, 1, 2, 3, 7, 6, 2, 3, 4, 2, 1, 5, 6, 3, 2, 3, 6, 2, 1.

Page faults: _____

☐ Least Recently Used (LRU) replacement

2, 1, 2, 3, 7, 6, 2, 3, 4, 2, 1, 5, 6, 3, 2, 3, 6, 2, 1.

Page faults: _____

☐ Optimal replacement

2, 1, 2, 3, 7, 6, 2, 3, 4, 2, 1, 5, 6, 3, 2, 3, 6, 2, 1.

Page faults: _____

การเขียนรายงานการทดลอง

1. ออกแบบการทำงานของโปรแกรมด้วยผังงาน (Flowchart)

อัด Clip video อธิบาย

2. อธิบายขั้นตอนการเขียนโปรแกรม พร้อม แคปเจอร์ รูปภาพขั้นตอนการทำงาน สรุปผลการทำงาน
3. อธิบายขั้นตอนการทำงานของ Algorithm ทั้ง 3 แบบ พร้อมวาดภาพประกอบโดยใช้ตัวเลขอ้างอิงตามการทดลองในการอธิบาย
4. ส่งโค้ดโปรแกรม พร้อมคำอธิบายโดยละเอียด
5. สรุปผลการทดลอง
6. ส่งไฟล์งานใน MS Teams

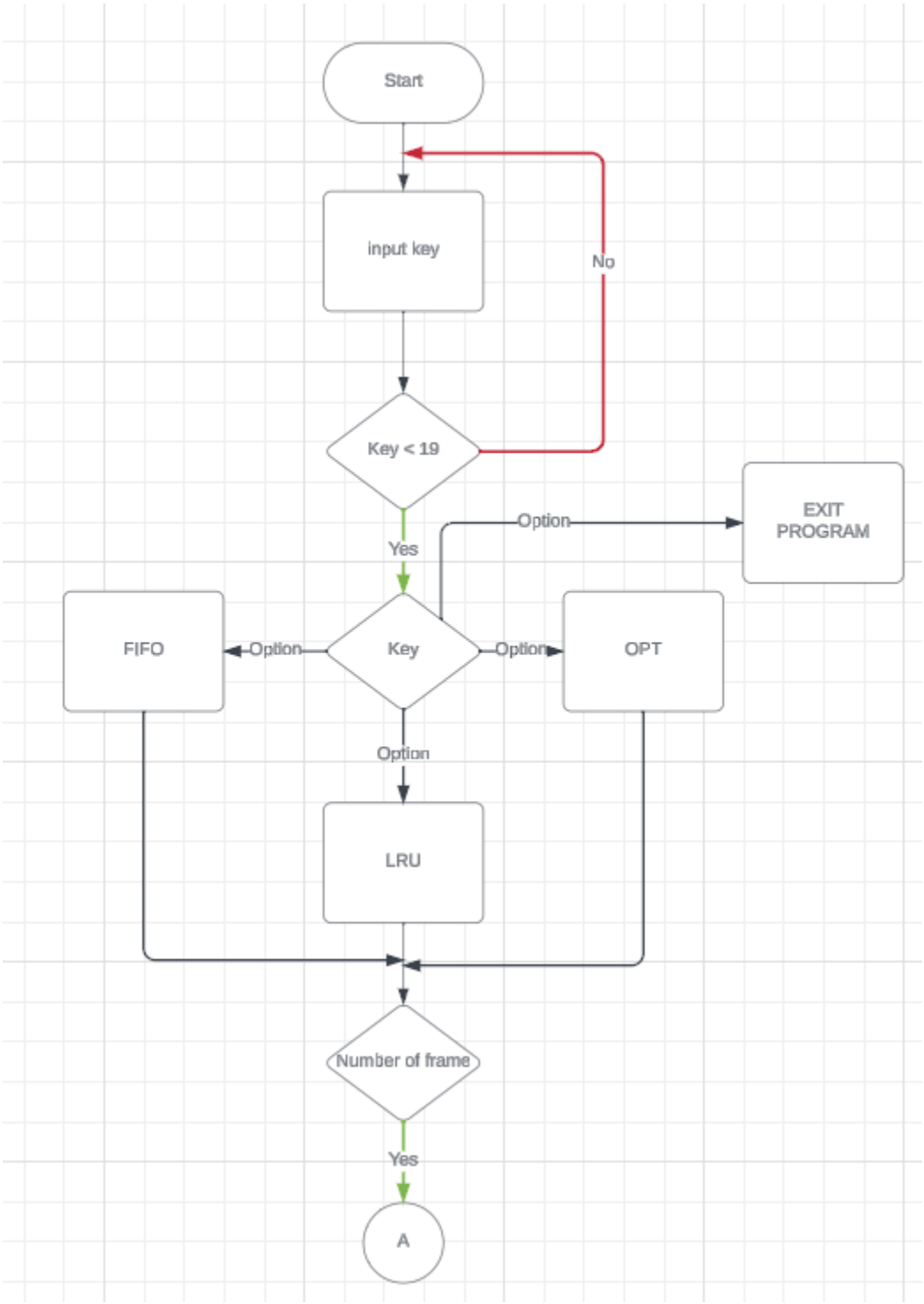
.....

.....

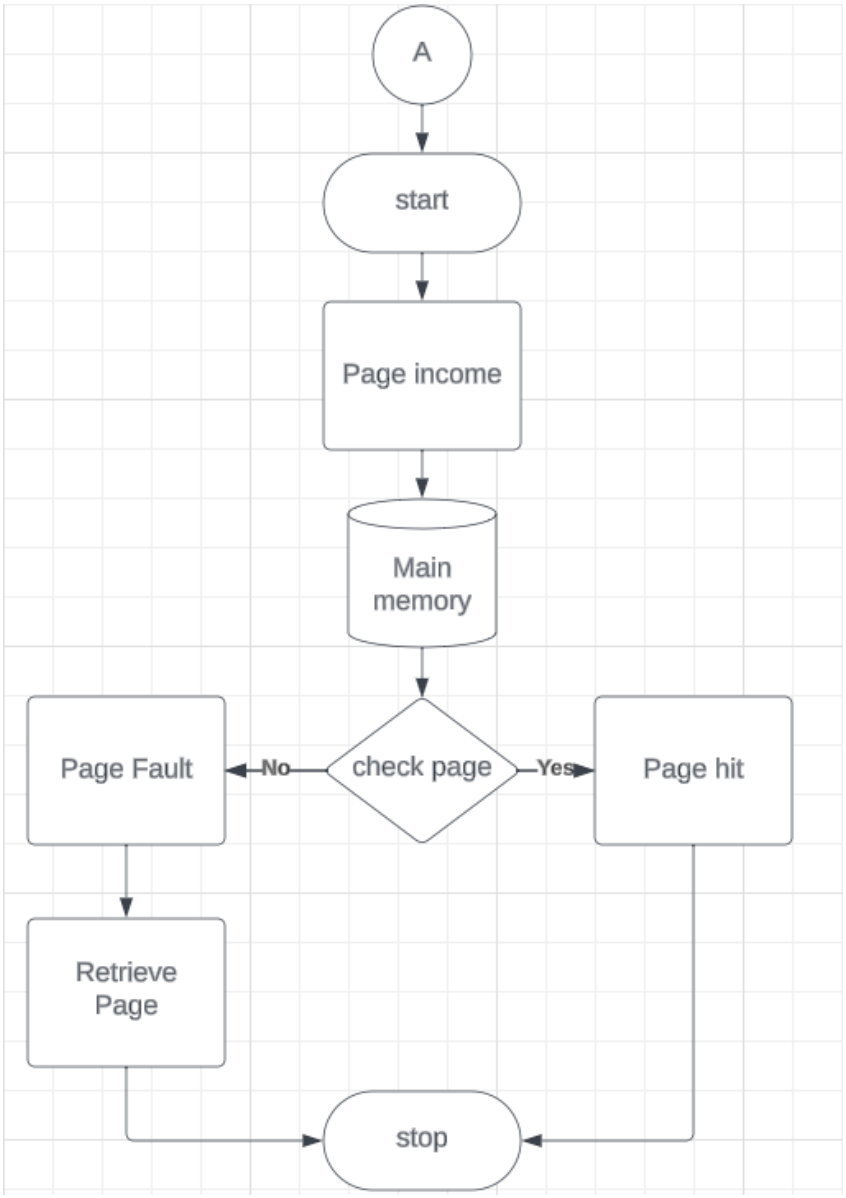
.....

.....

Flowchart Maincode

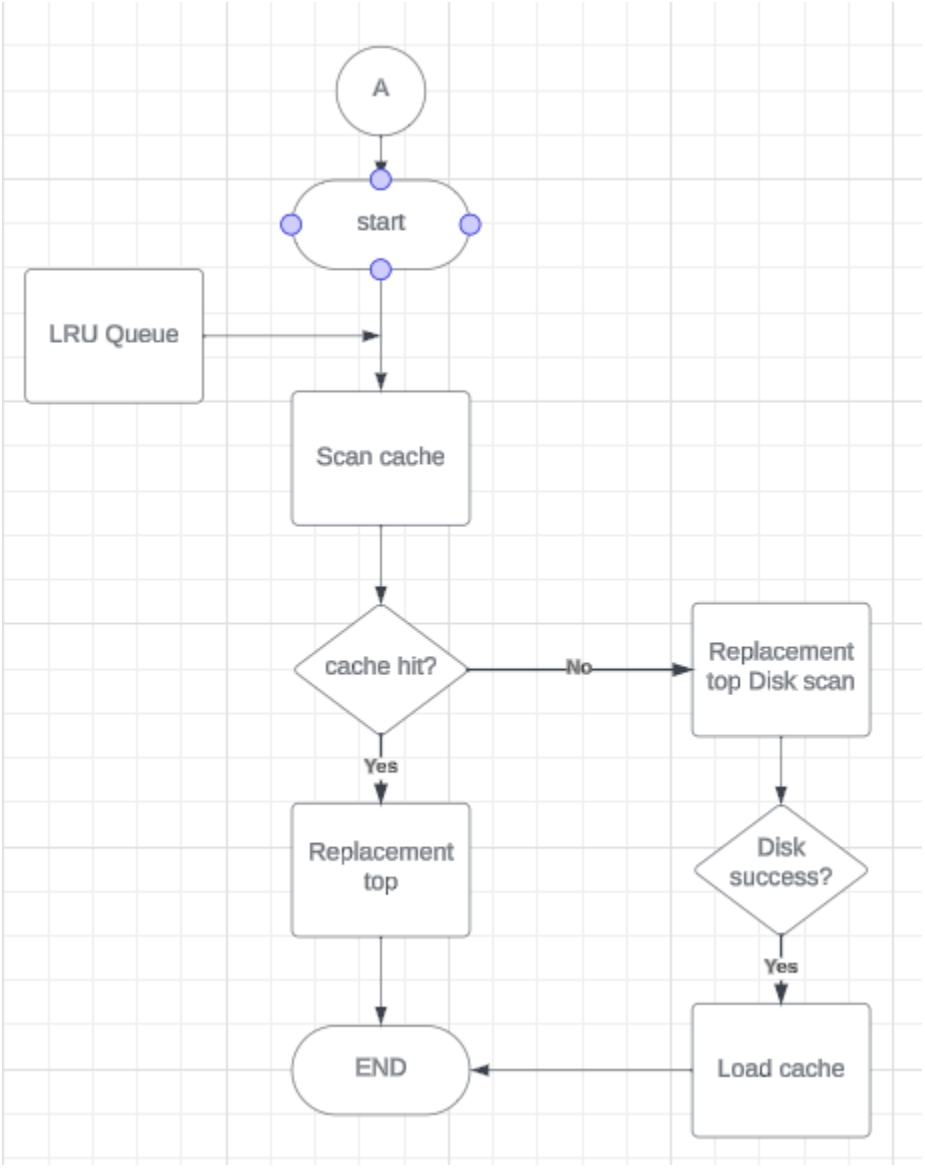


Flowchart FIFO



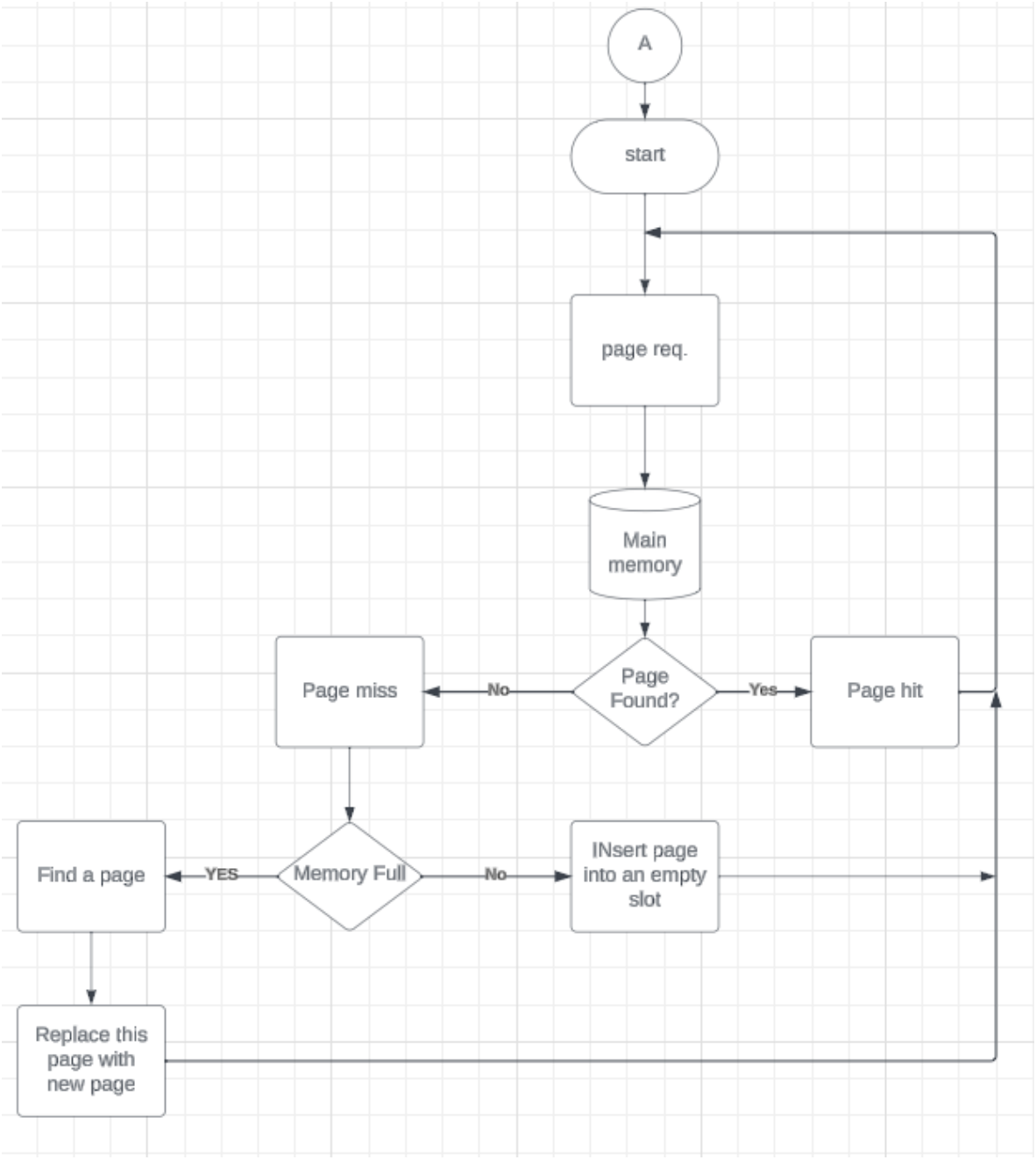
Reference string																			
	2	1	2	3	7	6	2	3	4	2	1	5	6	3	2	3	6	2	1
1. FIFO																			
	2	1	2	3	7	6	2	3	4	2	1	5	6	3	2	3	6	2	1
	2	2	2	2	2	6	6	6	6	6	6	5	5	5	5	5	5	5	1
		1	1	1	1	1	2	2	2	2	2	2	6	6	6	6	6	6	6
				3	3	3	3	3	4	4	4	4	4	3	3	3	3	3	3
					7	7	7	7	7	7	1	1	1	1	2	2	2	2	2
	F	F		F	F	F	F		F		F	F	F	F	F				F
	Page faults = 13																		

Flowchart LRU



2. LRU																			
2	1	2	3	7	6	2	3	4	2	1	5	6	3	2	3	6	2	1
2	2	1	1	1	2	3	7	6	6	3	4	2	1	5	5	5	5	3
	1	2	2	2	3	7	6	2	3	4	2	1	5	6	6	2	3	6
			3	3	7	6	2	3	4	2	1	5	6	3	2	3	6	2
				7	6	2	3	4	2	1	5	6	3	2	3	6	2	1
F	F		F	F	F			F		F	F	F	F	F				F
Page faults = 12																			

Flowchart OPT



3. OPT																			
2	1	2	3	7	6	2	3	4	2	1	5	6	3	2	3	6	2	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
			3	3	3	3	3	4	4	4	5	5	3	3	3	3	3	3
				7	6	6	6	6	6	6	6	6	6	6	6	6	6	6
F	F		F	F	F			F			F		F					
Page faults = 8																		

Coding

```c
struct Frame
{
    int page;
    int count;      //  FIFO
    int pastIndex;  //  LRU
    int nextIndex;  //  OPT
};

int page[19];
```

1. **ไฟล์ส่วนของ Header และการนิยาม Struct:**
   - โปรแกรมนำเข้าไฟล์ส่วนของภาษา C ที่จำเป็น (stdio.h สำหรับการนำเข้า/ส่งออก และ stdbool.h สำหรับการใช้งาน boolean)
   - นิยามโครงสร้างที่ชื่อ Frame เพื่อแทนตัวเฟรมหน้า (page frame) ในหน่วยความจำ มีฟิลด์สำหรับหมายเลขหน้า (page), ตัวนับสำหรับอัลกอริทึม FIFO (count), และดัชนีสำหรับอัลกอริทึม LRU (pastIndex) และ OPT (nextIndex)

2. **ตัวแปร Global:**
   - ประกาศอาร์เรย์ page[19] เพื่อเก็บหน้า 19 หน้าที่จะถูกนำมาใช้เป็นอินพุตสำหรับอัลกอริทึมการแทนหน้า

```
bool isValueInArray(int val, struct Frame frame[], int size);
int max(struct Frame frame[], int size, int key);
int min(struct Frame frame[], int size, int key);
int searchNextIndex(int Nframe, int nowIndex);
int searchPastIndex(int Nframe, int nowIndex);
int FIFO();
int LRU();
int OPT();
```

3. ฟังก์ชันประการตัวแบบ:

- ประกาศตัวแบบของฟังก์ชัน เช่น isValueInArray, max, min, searchNextIndex, searchPastIndex, และ 3 อัลกอริทึมหลัก FIFO, LRU, และ OPT

```c
int main()
{
    int key;
    printf("\n================ Page Replacement Program ==================\n");
    printf("Please input 19 pages : ");
    for (int i = 0; i < 19; i++)
    {
        scanf("%d", &page[i]);
    }
    printf("\nYour input : ");
    for (int i = 0; i < 19; i++)
    {
        printf("%d ", page[i]);
    }
    while (key != 4)
    {
        printf("\n\n================ Please select a menu ======================\n");
        printf("1. FIFO\n2. LRU\n3. OPT\n4. Exit Program\n");
        printf("Please input a key to continue : ");
        scanf("%d", &key);
        switch (key)
        {
        case 1:
            FIFO();
            break;
        case 2:
            LRU();
            break;
        case 3:
            OPT();
            break;
        }
    }
    return 0;
}
```

4. **การประกาศตัวแปร:**
   - int key ประกาศตัวแปรจำนวนเต็มชื่อ key เพื่อเก็บค่าที่ผู้ใช้เลือกจากเมนู.

5. **การรับข้อมูลจากผู้ใช้:**
   - โปรแกรมแสดงข้อความให้ผู้ใช้ป้อนหน้าข้อมูล 19 หน้า โดยใช้ลูป for และจัดเก็บข้อมูลที่ป้อนลงในอาร์เรย์ชื่อ page.

6. **การแสดงผลข้อมูลที่ผู้ใช้ป้อน:**
   - โปรแกรมแสดงผลข้อมูลที่ผู้ใช้ป้อนเพื่อยืนยันหน้าข้อมูลที่ป้อน.

7. **เมนูและลูป:**
   - โปรแกรมเข้าสู่ลูป while ซึ่งจะทำงานไปเรื่อยๆ จนกว่าผู้ใช้จะป้อนคีย์ 4 เพื่อออก.
   - ภายในลูป โปรแกรมแสดงเมนูที่ให้ผู้ใช้เลือกอัลกอริทึมการแทนหน้าที่ต่างกัน (FIFO, LRU, OPT, และ ปิดโปรแกรม).
   - ผู้ใช้ถูกแจ้งให้ป้อนคีย์และโปรแกรมใช้คำสั่ง switch เพื่อดำเนินการอัลกอริทึมที่ถูกเลือกตามค่าที่ผู้ใช้ป้อน.

8. **การทำงานของอัลกอริทึม:**
   - ถ้าผู้ใช้ป้อน 1 จะเรียกใช้ฟังก์ชัน FIFO()
   - ถ้าผู้ใช้ป้อน 2 จะเรียกใช้ฟังก์ชัน LRU()
   - ถ้าผู้ใช้ป้อน 3 จะเรียกใช้ฟังก์ชัน OPT()
   - ถ้าผู้ใช้ป้อน 4 จะทำการหยุดการทำงานของโปรแกรม

9. **คำสั่ง Return:**
   - ฟังก์ชัน main จะส่งค่า 0 เมื่อทำงานเสร็จสมบูรณ์.

```c
int FIFO(){
    int n = 0;
    int pf = 0;
    printf("Please input the number of frames : ");
    scanf("%d", &n);
    struct Frame frame[n]; // str frame
    for (int e = 0; e < n; e++){
        frame[e].page = -1;
        frame[e].count = 0;
    }
    for (int i = 0; i < 19; i++){
        if (!isValueInArray(page[i], frame, n)){// check value
            if (isValueInArray(-1, frame, n)){// check empty of arrays
                for (int e = 0; e < n; e++){
                    if (frame[e].page == -1){
                        frame[e].page = page[i];
                        frame[e].count++;
                        pf++;
                        break;
                    }else{
                        frame[e].count++;
                    }
                }
            }else{
                for (int e = 0; e < n; e++){
                    if (max(frame, n, 1) != e){
                        frame[e].count++;
                    }
                }
                frame[max(frame, n, 1)].page = page[i];
                frame[max(frame, n, 1)].count = 1;
                pf++;
            }
        }else{
            for (int e = 0; e < n; e++){
                frame[e].count++;
            }
        }
        printf("\nInput page : %d\n", page[i]);
        printf("Frame Content :\n");
        for (int e = 0; e < n; e++){
            if (frame[e].page == -1){
                printf("Frame %d : [ ]\n", e + 1);
            }else{
                printf("Frame %d : [%d]\n", e + 1, frame[e].page);
            }
        }
        printf("================================================================\n");
        printf("Page fault (FIFO) : %d\n", pf);
        printf("================================================================\n");
    }
    return 0;
}
```

10. การรับค่าจำนวนเฟรม (frames) จากผู้ใช้:

- โปรแกรมแสดงข้อความให้ผู้ใช้ป้อนจำนวนเฟรมที่ต้องการใช้.

11. การสร้างและเตรียมข้อมูลเฟรม:

- โปรแกรมสร้างอาร์เรย์ frame ที่มีขนาดเท่ากับจำนวนเฟรมที่ผู้ใช้ป้อน.
- แต่ละเฟรมมีสมาชิก page และ count โดยเริ่มต้น page ให้มีค่า -1 และ count ให้เป็น 0.

12. การประมวลผลหน้าข้อมูล:

- โปรแกรมทำการตรวจสอบว่าหน้าข้อมูลปัจจุบันอยู่ในเฟรมหรือไม่โดยใช้ฟังก์ชัน isValueInArray.
- ถ้าหน้าข้อมูลยังไม่อยู่ในเฟรม (page fault) โปรแกรมทำการตรวจสอบว่ามีเฟรมที่ว่างหรือไม่.
- ถ้ามีเฟรมว่าง โปรแกรมใส่หน้าข้อมูลลงในเฟรมที่ว่าง.
- ถ้าไม่มีเฟรมว่าง โปรแกรมจะเลือกเฟรมที่มีการใช้น้อยที่สุดในการแทนที่ (FIFO) และใส่หน้าข้อมูลลงไป.
- ถ้าหน้าข้อมูลมีอยู่ในเฟรม (ไม่เกิด page fault) โปรแกรมจะเพิ่มค่า count ในทุกๆ เฟรม.

13. การแสดงผลลัพธ์:

- หลังจากการประมวลผลแต่ละหน้าข้อมูล โปรแกรมทำการแสดงผลลัพธ์ที่แสดงหน้าข้อมูลที่ถูกป้อนเข้ามา, เนื้อหาของเฟรม, และจำนวน page fault ที่เกิดขึ้น.

14. คำสั่ง Return:

- ฟังก์ชัน FIFO ส่งค่า 0 เมื่อทำงานเสร็จสมบูรณ์.

```c
int LRU(){
    int n = 0;
    int pf = 0;
    printf("Please input the number of frames : ");
    scanf("%d", &n);
    struct Frame frame[n];
    for (int e = 0; e < n; e++){
        frame[e].page = -1;
        frame[e].pastIndex = 0;
    }
    for (int i = 0; i < 19; i++){
        if (!isValueInArray(page[i], frame, n)){
            if (isValueInArray(-1, frame, n)) {
                for (int e = 0; e < n; e++){
                    if (frame[e].page == -1){
                        frame[e].page = page[i];
                        frame[e].pastIndex = i;
                        pf++;
                        break;
                    }else{
                        frame[e].pastIndex = searchPastIndex(frame[e].page, i);
                    }
                }
            }else{
                frame[min(frame, n, 2)].page = page[i];
                frame[min(frame, n, 2)].pastIndex = searchPastIndex(frame[min(frame, n, 2)].page, i);
                pf++;
            }
        }else{
            for (int e = 0; e < n; e++){
                frame[e].pastIndex = searchPastIndex(frame[e].page, i);
            }
        }
        printf("\nInput page : %d\n", page[i]);
        printf("Frame Content :\n");
        for (int e = 0; e < n; e++){
            if (frame[e].page == -1){
                printf("Frame %d : [ ]\n", e + 1);
            }else{
                printf("Frame %d : [%d]\n", e + 1, frame[e].page);
            }
        }
        printf("============================================================\n");
        printf("Page fault (LRU) : %d\n", pf);
        printf("============================================================\n");
    }
    return 0;
}
```

15. การรับค่าจำนวนเฟรม (frames) จากผู้ใช้:

- โปรแกรมแสดงข้อความให้ผู้ใช้ป้อนจำนวนเฟรมที่ต้องการใช้.

16. การสร้างและเตรียมข้อมูลเฟรม:

- โปรแกรมสร้างอาร์เรย์ frame ที่มีขนาดเท่ากับจำนวนเฟรมที่ผู้ใช้ป้อน.
- แต่ละเฟรมมีสมาชิก page และ pastIndex โดยเริ่มต้น page ให้มีค่า -1 และ pastIndex ให้เป็น 0.

17. การประมวลผลหน้าข้อมูล:

- โปรแกรมทำการตรวจสอบว่าหน้าข้อมูลปัจจุบันอยู่ในเฟรมหรือไม่โดยใช้ฟังก์ชัน isValueInArray.
- ถ้าหน้าข้อมูลยังไม่อยู่ในเฟรม (page fault) โปรแกรมทำการตรวจสอบว่ามีเฟรมที่ว่างหรือไม่.
- ถ้ามีเฟรมว่าง โปรแกรมใส่หน้าข้อมูลลงในเฟรมที่ว่างพร้อมกับการเซ็ต pastIndex ในเฟรมนั้นเป็นค่าปัจจุบัน.
- ถ้าไม่มีเฟรมว่าง โปรแกรมจะเลือกเฟรมที่มีค่า pastIndex น้อยที่สุดในการแทนที่ (LRU) และใส่หน้าข้อมูลลงไปพร้อมกับการเซ็ต pastIndex ในเฟรมใหม่เป็นค่าปัจจุบัน.
- ถ้าหน้าข้อมูลมีอยู่ในเฟรม (ไม่เกิด page fault) โปรแกรมจะอัพเดตค่า pastIndex ในทุกๆ เฟรม.

18. การแสดงผลลัพธ์:

- หลังจากการประมวลผลแต่ละหน้าข้อมูล โปรแกรมทำการแสดงผลลัพธ์ที่แสดงหน้าข้อมูลที่ถูกป้อนเข้ามา, เนื้อหาของเฟรม, และจำนวน page fault ที่เกิดขึ้น.

19. คำสั่ง Return:

- ฟังก์ชัน LRU ส่งค่า 0 เมื่อทำงานเสร็จสมบ

```c
int OPT(){
    int n = 0;
    int pf = 0;
    printf("Please input the number of frames : ");
    scanf("%d", &n);
    struct Frame frame[n];
    for (int e = 0; e < n; e++){
        frame[e].page = -1;
        frame[e].nextIndex = 0;
    }
    for (int i = 0; i < 19; i++){// check value
        if (!isValueInArray(page[i], frame, n)) {
            if (isValueInArray(-1, frame, n)) {
                for (int e = 0; e < n; e++){
                    if (frame[e].page == -1){
                        frame[e].page = page[i];
                        frame[e].nextIndex = searchNextIndex(frame[e].page, i);
                        pf++;
                        break;
                    }else{
                        frame[e].nextIndex = searchNextIndex(frame[e].page, i);
                    }
                }
            }else{
                if (frame[min(frame, n, 1)].nextIndex == 0){
                    frame[min(frame, n, 1)].page = page[i];
                    frame[min(frame, n, 1)].nextIndex = searchNextIndex(frame[min(frame, n, 1)].page, i);
                }else{
                    frame[max(frame, n, 2)].page = page[i];
                    frame[max(frame, n, 2)].nextIndex = searchNextIndex(frame[max(frame, n, 2)].page, i);
                }
                pf++;
            }
        }else{
            for (int e = 0; e < n; e++){
                frame[e].nextIndex = searchNextIndex(frame[e].page, i);
            }
        }
        printf("\nInput page : %d\n", page[i]);
        printf("Frame Content :\n");
        for (int e = 0; e < n; e++){
            if (frame[e].page == -1){
                printf("Frame %d : [ ]\n", e + 1);
            }else{
                printf("Frame %d : [%d]\n", e + 1, frame[e].page);
            }
        }
```

```c
        printf("================================================================\n");
        printf("Page fault (OPT) : %d\n", pf);
        printf("================================================================\n");
    }
    return 0;
}
```

20. การรับค่าจำนวนเฟรม (frames) จากผู้ใช้:

- โปรแกรมแสดงข้อความให้ผู้ใช้ป้อนจำนวนเฟรมที่ต้องการใช้.

21. การสร้างและเตรียมข้อมูลเฟรม:

- โปรแกรมสร้างอาร์เรย์ frame ที่มีขนาดเท่ากับจำนวนเฟรมที่ผู้ใช้ป้อน.
- แต่ละเฟรมมีสมาชิก page และ nextIndex โดยเริ่มต้น page ให้มีค่า -1 และ nextIndex ให้เป็น 0.

22. การประมวลผลหน้าข้อมูล:

- โปรแกรมทำการตรวจสอบว่าหน้าข้อมูลปัจจุบันอยู่ในเฟรมหรือไม่โดยใช้ฟังก์ชัน isValueInArray.
- ถ้าหน้าข้อมูลยังไม่อยู่ในเฟรม (page fault) โปรแกรมทำการตรวจสอบว่ามีเฟรมที่ว่างหรือไม่.
- ถ้ามีเฟรมว่าง โปรแกรมใส่หน้าข้อมูลลงในเฟรมที่ว่างพร้อมกับการเซ็ต nextIndex ในเฟรมนั้นเป็นค่าที่บ่งบอกถึงการใช้งานต่อไปของหน้าข้อมูลนี้.
- ถ้าไม่มีเฟรมว่าง โปรแกรมจะเลือกเฟรมที่มีค่า nextIndex ที่มีการใช้งานล่าสุดน้อยที่สุด (Optimal) และใส่หน้าข้อมูลลงไปพร้อมกับการเซ็ต nextIndex ในเฟรมใหม่เป็นค่าที่บ่งบอกถึงการใช้งานต่อไปของหน้าข้อมูลนี้.
- ถ้าหน้าข้อมูลมีอยู่ในเฟรม (ไม่เกิด page fault) โปรแกรมจะอัพเดตค่า nextIndex ในทุกๆ เฟรม.

23. การแสดงผลลัพธ์:

- หลังจากการประมวลผลแต่ละหน้าข้อมูล โปรแกรมทำการแสดงผลลัพธ์ที่แสดงหน้าข้อมูลที่ถูกป้อนเข้ามา, เนื้อหาของเฟรม, และจำนวน page fault ที่เกิดขึ้น.

24. คำสั่ง Return:

- ฟังก์ชัน OPT ส่งค่า 0 เมื่อทำงานเสร็จสมบูรณ์.

```
bool isValueInArray(int val, struct Frame frame[], int size)
{
    for (int i = 0; i < size; i++)
    {
        if (frame[i].page == val)
            return true;
    }
    return false;
}
```

25. การรับค่า:

- รับค่า val ที่ต้องการตรวจสอบว่ามีอยู่ในอาร์เรย์หรือไม่.
- รับอาร์เรย์ของโครงสร้าง Frame ที่ชื่อ frame.
- รับขนาดของอาร์เรย์ size.

26. การทำงาน:

- ใช้ลูป for เพื่อวนลูปทุกรายการในอาร์เรย์ frame.

27. การเปรียบเทียบค่า:

- ที่แต่ละรอบของลูป for ฟังก์ชันตรวจสอบว่า frame[i].page เท่ากับค่า val หรือไม่.
- ถ้าเท่ากัน (ค่า val อยู่ในอาร์เรย์) ฟังก์ชันจะคืนค่า true.

28. คืนค่า:

- ถ้าไม่เจอค่า val ในทุกรายการในอาร์เรย์ frame ฟังก์ชันจะคืนค่า false.

```
int max(struct Frame frame[], int size, int key)
{
    int index = 0;
    if (key == 1)
    {
        int max = frame[index].count;
        for (int i = 0; i < size; i++)
        {
            if (frame[i].count > max)
            {
                max = frame[i].count;
                index = i;
            }
        }
    }
    else if (key == 2)
    {
        int max = frame[index].nextIndex;
        for (int i = 0; i < size; i++)
        {
            if (frame[i].nextIndex > max)
            {
                max = frame[i].nextIndex;
                index = i;
            }
        }
    }
    return index;
}
```

29. การรับค่า:

- รับอาร์เรย์ของโครงสร้าง Frame ที่ชื่อ frame.
- รับขนาดของอาร์เรย์ size.
- รับค่า key เพื่อระบุว่าจะใช้ค่า count หรือ nextIndex ในการค้นหา.

30. การคำนวณ:

- ตั้งค่าตัวแปร index เป็น 0 เพื่อเก็บดัชนีของค่าสูงสุด.
- ตั้งค่าตัวแปร max ในขอบเขตของลูป เพื่อให้สามารถเปรียบเทียบค่าได้.
- ในลูป for ทำการเปรียบเทียบค่าของ count หรือ nextIndex ของแต่ละเฟรม.
- หากค่าใดมีค่ามากกว่าค่า max ที่เก็บไว้, ก็ทำการอัพเดตค่า max และ index เป็นดัชนีของเฟรมนั้น.

31. คืนค่า:

- หลังจากลูปเสร็จสิ้น, ฟังก์ชันจะคืนค่า index ที่เก็บดัชนีของเฟรมที่มีค่าสูงสุด.

```
int min(struct Frame frame[], int size, int key)
{
    int index = 0;
    if (key == 1)
    {
        int min = frame[index].nextIndex;
        for (int i = 0; i < size; i++)
        {
            if (frame[i].nextIndex < min)
            {
                min = frame[i].nextIndex;
                index = i;
            }
        }
    }
    else if (key == 2)
    {
        int min = frame[index].pastIndex;
        for (int i = 0; i < size; i++)
        {
            if (frame[i].pastIndex < min)
            {
                min = frame[i].pastIndex;
                index = i;
            }
        }
    }
    return index;
}
```

32. การรับค่า:

- รับอาร์เรย์ของโครงสร้าง Frame ที่ชื่อ frame.
- รับขนาดของอาร์เรย์ size.
- รับค่า key เพื่อระบุว่าจะใช้ค่า nextIndex หรือ pastIndex ในการค้นหา.

33. การคำนวณ:

- ตั้งค่าตัวแปร index เป็น 0 เพื่อเก็บดัชนีของค่าน้อยที่สุด.
- ตั้งค่าตัวแปร min ในขอบเขตของลูป เพื่อให้สามารถเปรียบเทียบค่าได้.
- ในลูป for ทำการเปรียบเทียบค่าของ nextIndex หรือ pastIndex ของแต่ละเฟรม.
- หากค่าใดมีค่าน้อยกว่าค่า min ที่เก็บไว้, ก็ทำการอัพเดตค่า min และ index เป็นดัชนีของเฟรมนั้น.
-

34. คืนค่า:

- หลังจากลูปเสร็จสิ้น, ฟังก์ชันจะคืนค่า index ที่เก็บดัชนีของเฟรมที่มีค่าน้อยที่สุด.

```
int searchNextIndex(int Nframe, int nowIndex)
{
    int index = 0;
    for (int i = nowIndex + 1; i < 19; i++)
    {
        if (Nframe == page[i])
        {
            index = i;
            break;
        }
    }
    return index;
}
```

35. **การรับค่า:**

- รับค่า Nframe ซึ่งเป็นหน้าข้อมูลที่ต้องการค้นหาดัชนีของ.
- รับค่า nowIndex ซึ่งเป็นดัชนีปัจจุบันที่ต้องการเริ่มการค้นหา.

36. **การค้นหา:**

- ใช้ลูป for เพื่อวนลูปตั้งแต่ดัชนี nowIndex + 1 ไปจนถึงสุดท้ายของอาร์เรย์ page (จำนวนรอบที่เหลือเท่ากับ 19 - nowIndex - 1).
- ในแต่ละรอบของลูป, ทำการเปรียบเทียบค่าของ Nframe กับ page[i].
- หากค่าที่เรียกค้นหา (Nframe) ตรงกับค่าในอาร์เรย์ที่พบที่ดัชนี i, ก็ทำการกำหนดค่า index เป็น i และจบการทำงานของลูป.

37. **คืนค่า:**

- ฟังก์ชันคืนค่า index ที่เก็บดัชนีของหน้าข้อมูล Nframe ในอาร์เรย์ page ที่พบครั้งแรกหลังจาก nowIndex.

```
int searchPastIndex(int Nframe, int nowIndex)
{
    int index = 0;
    for (int i = nowIndex; i >= 0; i--)
    {
        if (Nframe == page[i])
        {
            index = i;
            break;
        }
    }
    return index;
}
```

38. การรับค่า:

- รับค่า Nframe ซึ่งเป็นหน้าข้อมูลที่ต้องการค้นหาดัชนีของ.
- รับค่า nowIndex ซึ่งเป็นดัชนีปัจจุบันที่ต้องการเริ่มการค้นหา.

39. การค้นหา:

- ใช้ลูป for เพื่อวนลูปตั้งแต่ดัชนี nowIndex ไปยัง 0.
- ในแต่ละรอบของลูป, ทำการเปรียบเทียบค่าของ Nframe กับ page[i].
- หากค่าที่เรียกค้นหา (Nframe) ตรงกับค่าในอาร์เรย์ที่พบที่ดัชนี i, ก็ทำการกำหนดค่า index เป็น i และจบการทำงานของลูป.

40. คืนค่า:

- ฟังก์ชันคืนค่า index ที่เก็บดัชนีของหน้าข้อมูล Nframe ในอาร์เรย์ page ที่พบครั้งแรกก่อน nowIndex.

สรุปผลการทดลอง

เมื่อโปรแกรมทำงานจะให้เรากรอกค่าจำนวนเลขที่เราต้องการที่จะกรอก และทำการกรอกเลข frame เพื่อนทำการประมวลผล และเรื่องระบบการทำงาน ว่าจะให้ทำงานแบบไหน