TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

FINAL EXAMINATION DATA STRUCTURES AND ALGORITHMS

Final examination report of Data Structures and Algorithms course

Người hướng dẫn: **TS. PHẠM THÁI KỲ TRUNG**

Người thực hiện: **PHẠM PHƯỚC TẤN – 520H0418**

Lớp : **20H50204**

Khoá : **24**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

FINAL EXAMINATION DATA STRUCTURES AND ALGORITHMS

Final examination report of Data Structures and Algorithms course

Người hướng dẫn: **TS. PHẠM THÁI KỲ TRUNG**
Người thực hiện: **PHẠM PHƯỚC TẤN**
Lớp : **20H50204**
Khoá : **24**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022

LỜI CẢM ƠN

I’m grateful to instruction and explanation of lecturer who is Pham Thai Ky Trung about the final report of this subject so I could effectively complete my report turning into better.

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi và được sự hướng dẫn của TS Phạm Thái Kỳ Trung. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 14 tháng 01 năm 2022

Tác giả

(ký tên và ghi rõ họ tên)

Phạm Phước Tấn

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

Tp. Hồ Chí Minh, ngày   tháng   năm
(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

Tp. Hồ Chí Minh, ngày   tháng   năm
(kí và ghi họ tên)

TÓM TẮT

In the final report of this subject so it has 5 problems which need to research including Recursion, Sorting, Stack, Data Structures, Graph. In this problems, I will solve following each step and explaining specifically for my report.

In problem 1 that is Recursion, I will draw the recursion tree and showing each step recursion in this problem and I was still implement the code.

In problem 2 that is Sorting, I will sort an array with ten elements and showing each step of it and in this problem I will choose 3 algorithms including Bubble, Selection and Quick sort.

In problem 3 that is Stack, I will analysis the infix which using stack to create a Reverse Polish Notation to calculate it to postfix notation and after having postfix notation, I will calculate it via syntax using stack and then I have the final result.

In problem 4 that is Data Structures, I will analysis three algorithms to managing students using Array, Linked List and an AVL tree to store the data and I will show the strength and weakness of three above algorithms with three operations and I will be implement the code of two algorithms including Linked List and AVL tree to calculate these operations.

In problem 5 that is Graph, I will analysis and draw the graph with 32 vertices and 55 edges of User and then, I will browse this graph with two algorithms including BFS and DFS. After this, I will be implement the code of this graph with three operations including reading a text file, finding the friend list of any vertex and couting the mutual friend of two any vertices and I will use the edge list to calculate it.

MỤC LỤC

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

CÁC KÝ HIỆU

CÁC CHỮ VIẾT TẮT

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH

DANH MỤC BẢNG

CHAPTER 1 – QUESTION 1: RECURSION

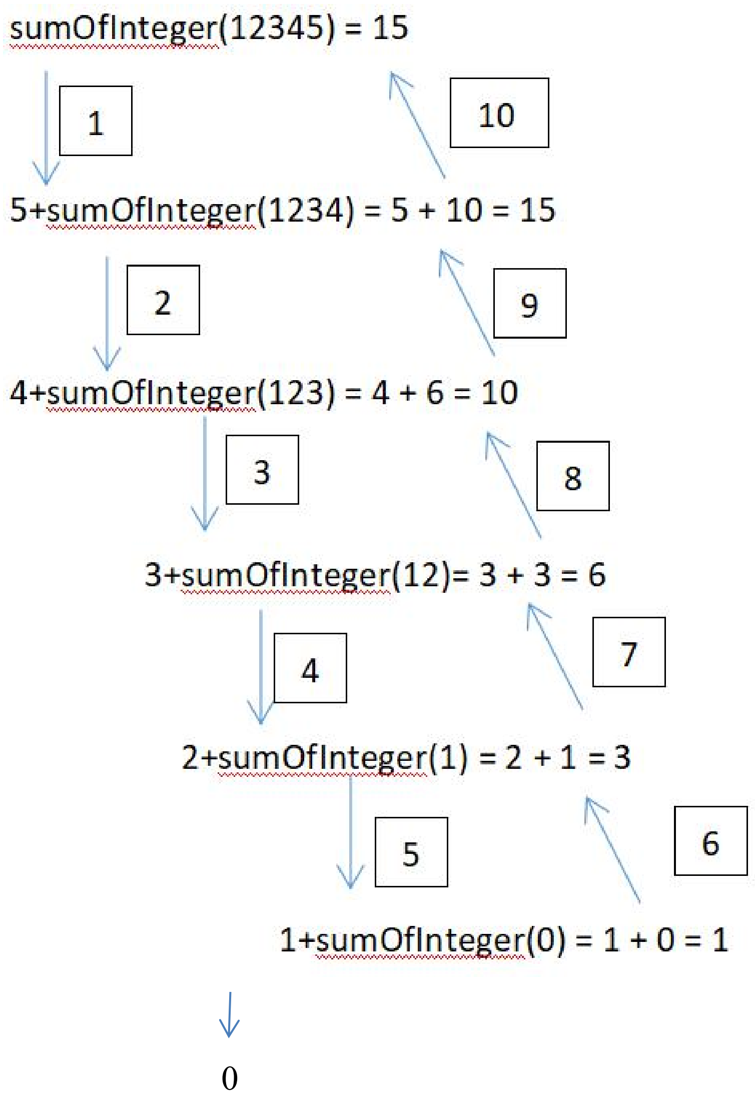
1.1 Theory

1.1.1 Group 1

I choose find the sum of all of the digits of a positive integer.

Suppose: Positive number is 12345.

I call method sumOfInteger(int n) with n = 12345 at main method so I will draw the recursion tree.


sumOfInteger(12345) = 15
1
10
5+sumOfInteger(1234) = 5 + 10 = 15
2
9
4+sumOfInteger(123) = 4 + 6 = 10
3
8
3+sumOfInteger(12)= 3 + 3 = 6
4
7
2+sumOfInteger(1) = 2 + 1 = 3
5
6
1+sumOfInteger(0) = 1 + 0 = 1
0

Picture 1.1.1 The recursion tree of group 1

First of all, I will called sumOfInteger(12345) method at main method then it will start recursion.

In step 1, with n =12345 to it will return n%10 + sumOfInteger(n/10) method so 12345%10 +sumOfInteger(12345/10) equal 5+ sumOfInteger(1234).

In step 2, it will continue with sumOfInteger(1234) method, due to n is different 0 so it will return 1234%10 +sumOfInteger(1234/10) equal 4+ sumOfInteger(123).

In step 3,it will continue with sumOfInteger(123) method, due to n is different 0 so it will return 123%10 +sumOfInteger(123/10) equal 3+ sumOfInteger(12).

In step 4,it will continue with sumOfInteger(12) method, due to n is different 0 so it will return 12%10 +sumOfInteger(12/10) equal 2+ sumOfInteger(1).

In step 5,it will continue with sumOfInteger(1) method, due to n is different 0 so it will return 1%10 +sumOfInteger(1/10) equal 1+ sumOfInteger(0) but sumOfInteger(0) due to n =0 so it return 0 and then it is 1+0 = 1.

In step 6, it will take the result of step 5 which is 1 to add with step 4 so 2+ sumOfInteger(1) = 2+1 = 3.

In step 7, it will take the result of step 6 which is 3 to add with step 3 so 3+ sumOfInteger(12) = 3+3 = 6.

In step 8, it will take the result of step 7 which is 6 to add with step 2 so 4+ sumOfInteger(123) = 4+6 = 10.

In step 9, it will take the result of step 8 which is 10 to add with step 1 so 5+ sumOfInteger(1234)= 5+10 = 15.

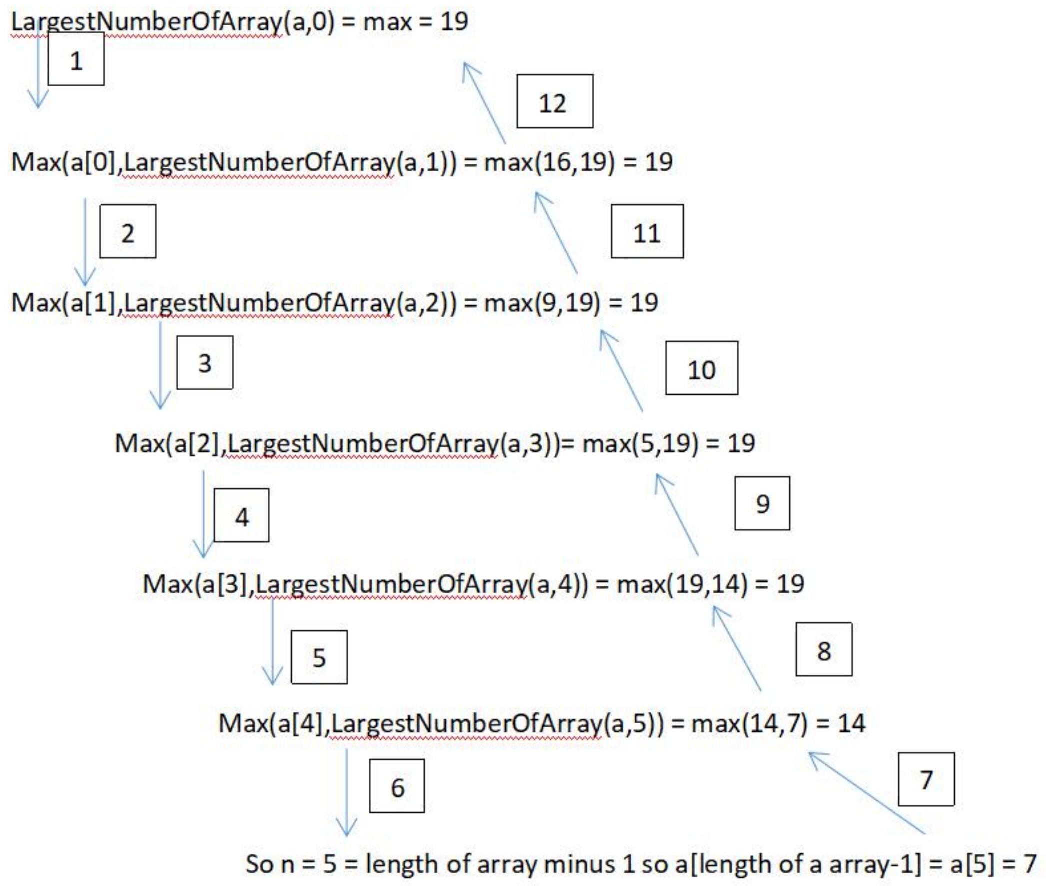
In the end step 10, this sum of all of the digits of a positive integer n = 12345 is 15.

1.1.2 Group 2

I choose find the largest number in an array.

Suppose: a =[16,9,5,19,14,7].

I call method LargestNumberOfArray(int[] a,int n) with a array and n=0 at main method so I will draw the recursion tree.


LargestNumberOfArray(a,0) = max = 19
1
12
Max(a[0],LargestNumberOfArray(a,1)) = max(16,19) = 19
2
11
Max(a[1],LargestNumberOfArray(a,2)) = max(9,19) = 19
3
10
Max(a[2],LargestNumberOfArray(a,3))= max(5,19) = 19
4
9
Max(a[3],LargestNumberOfArray(a,4)) = max(19,14) = 19
5
8
Max(a[4],LargestNumberOfArray(a,5)) = max(14,7) = 14
6
7
So n = 5 = length of array minus 1 so a[length of a array-1] = a[5] = 7

Picture 1.1.2 The recursion tree of group 2

Firstly, I will called LargestNumberOfArray(a,n) method at main method with array is a =[16,9,5,19,14,7] and n = 0 then it will start recursion with LargestNumberOfArray(a,0) method.

In step 1, with length of the a array is 6, due to n = 0 so it isn't equal length of the a array minus 1 so it will return max(a[n], LargestNumberOfArray(a,n+1)) then max(a[0], LargestNumberOfArray(a,0+1)) so it will be max (a[0],LargestNumberOfArray(a,1)) becoming max (16,LargestNumberOfArray(a,1)).

In step 2, it will be continue with LargestNumberOfArray(a,1) with length of the a array is 6, due to n = 1 so it isn't equal length of the a array minus 1 so it will return max(a[n], LargestNumberOfArray(a,n+1)) so it will be max (a[1],LargestNumberOfArray(a,2)) becoming max (9,LargestNumberOfArray(a,2)).

In step 3, it will be continue with LargestNumberOfArray(a,2) with length of the a array is 6, due to n = 2 so it isn't equal length of the a array minus 1 so it will return max(a[n], LargestNumberOfArray(a,n+1)) so it will be max (a[2],LargestNumberOfArray(a,3)) becoming max (5,LargestNumberOfArray(a,3)).

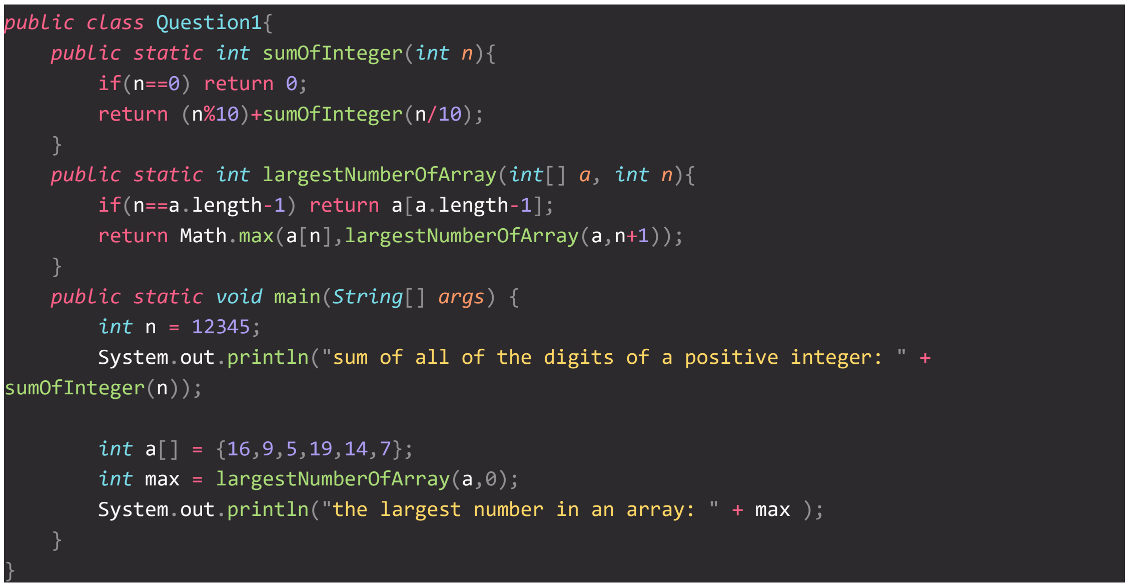
In step 4, it will be continue with LargestNumberOfArray(a,3) with length of the a array is 6, due to n = 3 so it isn't equal length of the a array minus 1 so it will return max(a[n], LargestNumberOfArray(a,n+1)) so it will be max (a[3],LargestNumberOfArray(a,4)) becoming max (19,LargestNumberOfArray(a,4)).

In step 5, it will be continue with LargestNumberOfArray(a,4) with length of the a array is 6, due to n = 4 so it isn't equal length of the a array minus 1 so it will return max(a[n], LargestNumberOfArray(a,n+1)) so it will be max (a[4],LargestNumberOfArray(a,5)) becoming max (14,LargestNumberOfArray(a,5)).

In step 6, it will be continue with LargestNumberOfArray(a,5) with length of the a array is 6, due to n = 5 so it is equal length of the a array minus 1. Therefore, it will return a[length of the a array minus 1] = a[5] = 7 so LargestNumberOfArray(a,5) equal 7.

In step 7, I will take the result of step 6 which is 7. After that, I will compare with step 5 , there is max(14,LargestNumberOfArray(a,5)) becoming max (14,7) = 14 so max(14,LargestNumberOfArray(a,5)) = 14 of LargestNumberOfArray (a,4).
In step 8, I will take the result of step 7 which is LargestNumberOfArray (a,4) = 14. After that, I will compare with step 4 , there is max (19,LargestNumberOfArray(a,4)) becoming max (19,14) = 19 so max(19,LargestNumberOfArray(a,4)) = 19 of LargestNumberOfArray(a,3).
In step 9, I will take the result of step 8 which is LargestNumberOfArray (a,3) = 19. After that, I will compare with step 3 , there is max (5,LargestNumberOfArray(a,3)) becoming max (5,19) = 19 so max(5,LargestNumberOfArray(a,3)) = 19 of LargestNumberOfArray(a,2).
In step 10, I will take the result of step 9 which is LargestNumberOfArray (a,2) = 19. After that, I will compare with step 2 , there is max (9,LargestNumberOfArray(a,2)) becoming max (9,19) = 19 so max(9,LargestNumberOfArray(a,2)) = 19 of LargestNumberOfArray(a,1).
In step 11, I will take the result of step 10 which is LargestNumberOfArray (a,1) = 19. After that, I will compare with step 1 , there is max (16,LargestNumberOfArray(a,1)) becoming max (16,19) = 19 so max(16,LargestNumberOfArray(a,1)) = 19 of LargestNumberOfArray(a,0).
In the end step, with LargestNumberOfArray(a,0) = 19 so the a array is empty element so it finished comparing and final, the largest number in array a is 19 value.

1.2 Practical



```
public class Question1{
    public static int sumOfInteger(int n){
        if(n==0) return 0;
        return (n%10)+sumOfInteger(n/10);
    }
    public static int largestNumberOfArray(int[] a, int n){
        if(n==a.length-1) return a[a.length-1];
        return Math.max(a[n],largestNumberOfArray(a,n+1));
    }
    public static void main(String[] args) {
        int n = 12345;
        System.out.println("sum of all of the digits of a positive integer: " +
sumOfInteger(n));

        int a[] = {16,9,5,19,14,7};
        int max = largestNumberOfArray(a,0);
        System.out.println("the largest number in an array: " + max );
    }
}
```

CHAPTER 2 – QUESTION 2: SORTING

2.1 Theory a

The algorithms are Bubble sort and Selection sort.

I have an array which is A[10]=[13,17,15,4,2,9,7,6,10,1]

2.1.1 Bubble sort

The first sort is Bubble sort.I want to an array being ascending. I will compare two elements next to each others and then if the first element is bigger than the behind element then I will swap two elements.By doing it repeatedly to the last position on the list.Until after n − 1 passes the list is sorted.

With the first for loop is index i = 0 and browse until fewer than the length of an array minus 1 then will finish so after each for loop , it will increase 1 unit.In i loop, I will create a for loop with index j = 0 browsing until fewer than the length of an array minus 1 and minus index i then will finish so after each for loop , it will increase 1 unit.In j for loop is comparing two elements which is between of two elements of index j and index j+1 so if element of j > element of j+1 so swap between of two elements of j and j+1 , if not, then it will keep going the continue j for loop.

The below array A, it don't sorting, it original.

13	17	15	4	2	9	7	6	10	1

Step 1: i=0

The first j loop, with j=0, I will compare between j with j+1, due A[j] = A[0] =13 and A[j+1] =A[1] = 17 so it won't swap.

13	17	15	4	2	9	7	6	10	1

Continue the j loop, with j=1, I will compare between j with j+1, due A[j] = A[1] =17 and A[j+1] =A[2] = 15 so A[1] > A[2] due 17 > 15 so it will swap.

13	15	17	4	2	9	7	6	10	1

Continue the j loop, with j=2, I will compare between j with j+1, due A[j] = A[2] =17 and A[j+1] =A[3] = 4 so A[2] > A[3] due 17 > 4 so it will swap.

13	15	4	17	2	9	7	6	10	1

Continue the j loop, with j=3, I will compare between j with j+1, due A[j] = A[3] =17 and A[j+1] =A[4] = 2 so A[3] > A[4] due 17 > 2 so it will swap.

13	15	4	2	17	9	7	6	10	1

Continue the j loop, with j=4, I will compare between j with j+1, due A[j] = A[4] =17 and A[j+1] =A[5] = 9 so A[4] > A[5] due 17 > 9 so it will swap.

13	15	4	2	9	17	7	6	10	1

Continue the j loop, with j=5, I will compare between j with j+1, due A[j] = A[5] =17 and A[j+1] =A[6] = 7 so A[5] > A[6] due 17 > 7 so it will swap.

13	15	4	2	9	7	17	6	10	1

Continue the j loop, with j=6, I will compare between j with j+1, due A[j] = A[6] =17 and A[j+1] =A[7] = 6 so A[6] > A[7] due 17 > 6 so it will swap.

13	15	4	2	9	7	6	17	10	1

Continue the j loop, with j=7, I will compare between j with j+1, due A[j] = A[7] =17 and A[j+1] =A[8] = 10 so A[7] > A[8] due 17 >10 so it will swap.

13	15	4	2	9	7	6	10	17	1

Continue the j loop, with j=8, I will compare between j with j+1, due A[j] = A[8] =17 and A[j+1] =A[9] = 10 so A[8] > A[9] due 17 >1 so it will swap.

13	15	4	2	9	7	6	10	1	17

Due index j equal the length of array A minus 1 minus index I then length of array a = 9-1-0 = 8 so it will finish the j for loop. And continue step 2.

Step 2: i=1

The first j loop, with j=0, I will compare between j with j+1, due A[j] = A[0] =13 and A[j+1] =A[1] = 15, A[1] < A[2] due 13 < 15 so it won't swap.

13	15	4	2	9	7	6	10	1	17

Continue the j loop, with j=1, I will compare between j with j+1, due A[j] = A[1] =15 and A[j+1] =A[2] = 4 so A[1] > A[2] due 15 > 4 so it will swap.

13	4	15	2	9	7	6	10	1	17

Continue the j loop, with j=2, I will compare between j with j+1, due A[j] = A[2] =15 and A[j+1] =A[3] = 2 so A[2] > A[3] due 15 > 2 so it will swap.

13	4	2	15	9	7	6	10	1	17

Continue the j loop, with j=3, I will compare between j with j+1, due A[j] = A[3] =15 and A[j+1] =A[4] = 9 so A[3] > A[4] due 15 > 9 so it will swap.

13	4	2	9	15	7	6	10	1	17

Continue the j loop, with j=4, I will compare between j with j+1, due A[j] = A[4] =15 and A[j+1] =A[5] = 7 so A[4] > A[5] due 15 > 7 so it will swap.

13	4	2	9	7	15	6	10	1	17

Continue the j loop, with j=5, I will compare between j with j+1, due A[j] = A[5] =15 and A[j+1] =A[6] = 6 so A[5] > A[6] due 15 > 6 so it will swap.

13	4	2	9	7	6	15	10	1	17

Continue the j loop, with j=6, I will compare between j with j+1, due A[j] = A[6] =15 and A[j+1] =A[7] = 10 so A[6] > A[7] due 15 > 10 so it will swap.

13	4	2	9	7	6	10	15	1	17

Continue the j loop, with j=7, I will compare between j with j+1, due A[j] = A[7] =15 and A[j+1] =A[8] = 1 so A[7] > A[8] due 15 >1 so it will swap.

13	4	2	9	7	6	10	1	15	17

Due index j equal the length of array A minus 1 minus index i with i = 1 then length of array a = 9 - 1 -1 = 7 so it will finish the j for loop. And continue step 3.

Step 3: i=2

The first j loop, with j=0, I will compare between j with j+1, due A[j] = A[0] =13 and A[j+1] =A[1] = 4, A[1] > A[2] due 13 >4 so it will swap.

4	13	2	9	7	6	10	1	15	17

Continue the j loop, with j=1, I will compare between j with j+1, due A[j] = A[1] =13 and A[j+1] =A[2] = 2 so A[1] > A[2] due 13 > 2 so it will swap.

4	2	13	9	7	6	10	1	15	17

Continue the j loop, with j=2, I will compare between j with j+1, due A[j] = A[2] =13 and A[j+1] =A[3] = 9 so A[2] > A[3] due 13> 9 so it will swap.

4	2	9	13	7	6	10	1	15	17

Continue the j loop, with j=3, I will compare between j with j+1, due A[j] = A[3] =13 and A[j+1] =A[4] = 7 so A[3] > A[4] due 13 > 7 so it will swap.

4	2	9	7	13	6	10	1	15	17

Continue the j loop, with j=4, I will compare between j with j+1, due A[j] = A[4] =13 and A[j+1] =A[5] = 6 so A[4] > A[5] due 13> 6 so it will swap.

4	2	9	7	6	13	10	1	15	17

Continue the j loop, with j=5, I will compare between j with j+1, due A[j] = A[5] =13 and A[j+1] =A[6] = 10 so A[5] > A[6] due 13 > 10 so it will swap.

4	2	9	7	6	10	13	1	15	17

Continue the j loop, with j=6, I will compare between j with j+1, due A[j] = A[6] =13 and A[j+1] =A[7] = 1 so A[6] > A[7] due 13 > 1 so it will swap.

4	2	9	7	6	10	1	13	15	17

Due index j equal the length of array A minus 1 minus index i with i = 2 then length of array a = 9 - 1 -2 = 6 so it will finish the j for loop. And continue step 4.

Step 4: i=3

The first j loop, with j=0, I will compare between j with j+1, due A[j] = A[0] =4 and A[j+1] =A[1] = 2, A[1] > A[2] due 4 >2 so it will swap.

2	4	9	7	6	10	1	13	15	17

Continue the j loop, with j=1, I will compare between j with j+1, due A[j] = A[1] =4 and A[j+1] =A[2] = 9 so A[1] < A[2] due 4 < 9 so it won't swap.

2	4	9	7	6	10	1	13	15	17

Continue the j loop, with j=2, I will compare between j with j+1, due A[j] = A[2] =9 and A[j+1] =A[3] = 7 so A[2] > A[3] due 9> 7 so it will swap.

2	4	7	9	6	10	1	13	15	17

Continue the j loop, with j=3, I will compare between j with j+1, due A[j] = A[3] =9 and A[j+1] =A[4] = 6 so A[3] > A[4] due 9 > 6 so it will swap.

2	4	7	6	9	10	1	13	15	17

Continue the j loop, with j=4, I will compare between j with j+1, due A[j] = A[4] =9 and A[j+1] =A[5] = 10 so A[4] < A[5] due 9< 10 so it won't swap.

2	4	7	6	9	10	1	13	15	17

Continue the j loop, with j=5, I will compare between j with j+1, due A[j] = A[5] =10 and A[j+1] =A[6] = 1 so A[5] > A[6] due 10 > 1 so it will swap.

2	4	7	6	9	1	10	13	15	17

Due index j equal the length of array A minus 1 minus index i with i = 3 then length of array a = 9 - 1 -3 = 5 so it will finish the j for loop. And continue step 5.

Step 5: i=4

The first j loop, with j=0, I will compare between j with j+1, due A[j] = A[0] =2 and A[j+1] =A[1] = 4, A[1] < A[2] due 2 <4 so it won't swap.

2	4	7	6	9	1	10	13	15	17

Continue the j loop, with j=1, I will compare between j with j+1, due A[j] = A[1] =4 and A[j+1] =A[2] = 7 so A[1] < A[2] due 4 < 7 so it won't swap.

2	4	7	6	9	1	10	13	15	17

Continue the j loop, with j=2, I will compare between j with j+1, due A[j] = A[2] =7 and A[j+1] =A[3] = 6 so A[2] > A[3] due 7> 6 so it will swap.

2	4	6	7	9	1	10	13	15	17

Continue the j loop, with j=3, I will compare between j with j+1, due A[j] = A[3] =7 and A[j+1] =A[4] = 9 so A[3] < A[4] due 7 < 9 so it won't swap.

2	4	6	7	9	1	10	13	15	17

Continue the j loop, with j=4, I will compare between j with j+1, due A[j] = A[4] =9 and A[j+1] =A[5] = 1 so A[4] > A[5] due 9 > 1 so it will swap.

2	4	6	7	1	9	10	13	15	17

Due index j equal the length of array A minus 1 minus index i with i = 4 then length of array a = 9 - 1 -4 = 4 so it will finish the j for loop. And continue step 6.

Step 6: i=5

The first j loop, with j=0, I will compare between j with j+1, due A[j] = A[0] =2 and A[j+1] =A[1] = 4, A[1] < A[2] due 2 <4 so it won't swap.

2	4	6	7	1	9	10	13	15	17

Continue the j loop, with j=1, I will compare between j with j+1, due A[j] = A[1] =4 and A[j+1] =A[2] = 6 so A[1] < A[2] due 4 < 6 so it won't swap.

2	4	6	7	1	9	10	13	15	17

Continue the j loop, with j=2, I will compare between j with j+1, due A[j] = A[2] =6 and A[j+1] =A[3] = 7 so A[2] < A[3] due 6 < 7 so it won't swap.

2	4	6	7	1	9	10	13	15	17

Continue the j loop, with j=3, I will compare between j with j+1, due A[j] = A[3] =7 and A[j+1] =A[4] = 1 so A[3] > A[4] due 7 > 1 so it will swap.

2	4	6	1	7	9	10	13	15	17

Due index j equal the length of array A minus 1 minus index i with i = 5 then length of array a = 9 - 1 -5 = 3so it will finish the j for loop. And continue step 7.

Step 7: i=6

The first j loop, with j=0, I will compare between j with j+1, due A[j] = A[0] =2 and A[j+1] =A[1] = 4, A[1] < A[2] due 2 <4 so it won't swap.

2	4	6	1	7	9	10	13	15	17

Continue the j loop, with j=1, I will compare between j with j+1, due A[j] = A[1] =4 and A[j+1] =A[2] = 6 so A[1] < A[2] due 4 < 6 so it won't swap.

2	4	6	1	7	9	10	13	15	17

Continue the j loop, with j=2, I will compare between j with j+1, due A[j] = A[2] =6 and A[j+1] =A[3] = 1 so A[2] > A[3] due 6 > 1 so it will swap.

2	4	1	6	7	9	10	13	15	17

Due index j equal the length of array A minus 1 minus index i with i = 6 then length of array a = 9 - 1 -6 = 2 so it will finish the j for loop. And continue step 8.

Step 8: i=7

The first j loop, with j=0, I will compare between j with j+1, due A[j] = A[0] =2 and A[j+1] =A[1] = 4, A[1] < A[2] due 2 <4 so it won't swap.

2	4	1	6	7	9	10	13	15	17

Continue the j loop, with j=1, I will compare between j with j+1, due A[j] = A[1] =4 and A[j+1] =A[2] = 6 so A[1] > A[2] due 4 > 1 so it will swap.

2	1	4	6	7	9	10	13	15	17

Due index j equal the length of array A minus 1 minus index i with i = 7 then length of array a = 9 - 1 -7 = 1 so it will finish the j for loop. And continue step 9.

Step 9: i=8

The first j loop, with j=0, I will compare between j with j+1, due A[j] = A[0] =2 and A[j+1] =A[1] = 1, A[1] > A[2] due 2 > 1 so it will swap.

1	2	4	6	7	9	10	13	15	17

Due index j equal the length of array A minus 1 minus index i with i = 8 then length of array a = 9 - 1 -8 = 0 so it will finish the j for loop. And finish the for j loop so also the i loop because i<n - 1 with n is the length of array A then array A has the length which is 9 so n-1 = 9-1 = 8.Therefore, it finish at here.

Finally, we have an array A which is sorted ascending below :

1	2	4	6	7	9	10	13	15	17

2.1.2 Selection sort

The Selection sort is kind of sort which is looking for the biggest or smallest a value so I just need to search an element which has the biggest or smallest a value then I will compare each element in array and after this, I will swap between of it and placing it in the correct position of the array.
With the first for loop is index i = 0 and browse until fewer than the length of an array minus 1 then will finish so after each for loop , it will increase 1 unit.In i loop, I will create a min variable because I will sort this ascending so I will assign the value i to min variable.after this, I will will create a for loop with index j = index i add 1 browsing until fewer than the length of an array, then will finish so after each for loop , it will increase 1 unit.In j for loop is comparing two element which is min and j value so if element of min bigger than element of j so will assign index j to min and then browsing until the last index of array.Finish the for j loop, then I will swap between of the element of min and element of i and continue browsing index i until the last element of array.
The below array A, it don't sorting, it original.

13	17	15	4	2	9	7	6	10	1

Step 1: with i =0, min =i then min = 0
The first j loop, with j=i+1 = 0+1 = 1, I will compare between element of j and element of min, due A[j] = A[1] =17 and A[min] =A[0] = 13, A[min] < A[1] due 13 < 17 so it won't change between of index j and min.

With j loop, index j will increase 1 unit so index j = 2, I will compare between element of j and element of min, due A[j] = A[2] =15 and A[min] =A[0] = 13, A[0] < A[2] due 13 < 15 so it won't change between of index j and min.

With j loop, index j will increase 1 unit so index j = 3, I will compare between element of j and element of min, due A[j] = A[3] =4 and A[min] =A[0] = 13, A[0] > A[3] due 13 > 4 so it will change between of index j and min so index min = 3.

With j loop, index j will increase 1 unit so index j = 4, I will compare between element of j and element of min, due A[j] = A[4] = 2 and A[min] =A[3] = 4, A[3] > A[4] due 4 > 2 so it will change between of index j and min so index min = 4.

With j loop, index j will increase 1 unit so index j = 5, I will compare between element of j and element of min, due A[j] = A[5] = 9 and A[min] =A[4] = 2, A[4] < A[5] due 2 < 9 so it won't change between of index j and min.

With j loop, index j will increase 1 unit so index j = 6, I will compare between element of j and element of min, due A[j] = A[6] = 7 and A[min] =A[4] = 2, A[4] < A[6] due 2 < 7 so it won't change between of index j and min.

With j loop, index j will increase 1 unit so index j = 5, I will compare between element of j and element of min, due A[j] = A[5] = 9 and A[min] =A[4] = 2, A[4] < A[5] due 2 < 9 so it won't change between of index j and min.

With j loop, index j will increase 1 unit so index j = 7, I will compare between element of j and element of min, due A[j] = A[7] = 6 and A[min] =A[4] = 2, A[4] < A[7] due 2 < 6 so it won't change between of index j and min.

With j loop, index j will increase 1 unit so index j = 8, I will compare between element of j and element of min, due A[j] = A[8] = 10 and A[min] =A[4] = 2, A[4] < A[8] due 2 < 10 so it won't change between of index j and min.

With j loop, index j will increase 1 unit so index j = 9, I will compare between element of j and element of min, due A[j] = A[9] = 1 and A[min] =A[4] = 2, A[4] > A[9] due 2 > 1 so it will change between of index j and min so index min = 9.

Due index j just browse until fewer than length of array A so j just run at j =9 after that, finish the j loop. I will swap between of the element of index min with element of i so swap A[min] = a[9] and A[i] = A[0] so array A will change below:

1	17	15	4	2	9	7	6	10	13

Step 2: with i =1, min =i then min = 1
The first j loop, with j=i+1 = 1+1 = 2, I will compare between element of j and element of min, due A[j] = A[2] =15 and A[min] =A[1] = 17, A[1] > A[2] due 17 > 15 so it will change between of index j and min so index min = j = 2.
With j loop, index j will increase 1 unit so index j = 3, I will compare between element of j and element of min, due A[j] = A[3] =4 and A[min] =A[2] = 15, A[2] > A[3] due 15> 4 so it will change between of index j and min so index min = j = 3.
With j loop, index j will increase 1 unit so index j = 4, I will compare between element of j and element of min, due A[j] = A[4] =2 and A[min] =A[3] = 4, A[3] > A[4] due 4> 2 so it will change between of index j and min so index min = j = 4.
With j loop, index j will increase 1 unit so index j = 5, I will compare between element of j and element of min, due A[j] = A[5] =9 and A[min] =A[4] = 2, A[4] < A[5] due 2 < 9 so it won’t change between of index j and min.
With j loop, index j will increase 1 unit so index j = 6, I will compare between element of j and element of min, due A[j] = A[6] = 7 and A[min] =A[4] = 2, A[4] < A[6] due 2 < 7 so it won’t change between of index j and min.
With j loop, index j will increase 1 unit so index j = 7, I will compare between element of j and element of min, due A[j] = A[7] = 6 and A[min] =A[4] = 2, A[4] < A[7] due 2 < 6 so it won’t change between of index j and min.
With j loop, index j will increase 1 unit so index j = 8, I will compare between element of j and element of min, due A[j] = A[8] =10 and A[min] =A[4] = 2, A[4] < A[8] due 2 < 10 so it won’t change between of index j and min.

With j loop, index j will increase 1 unit so index j = 9, I will compare between element of j and element of min, due A[j] = A[9] =13 and A[min] =A[4] = 2, A[4] < A[9] due 2 < 13 so it won't change between of index j and min.
Due index j just browse until fewer than length of array A so j just run at j =9 after that, finish the j loop. I will swap between of the element of index min with element of i so swap A[min] = a[4] and A[i] = A[1] so array A will change below:

1	2	15	4	17	9	7	6	10	13

Step 3: with i =2, min =i then min = 2
The first j loop, with j=i+1 = 2+1 = 3, I will compare between element of j and element of min, due A[j] = A[3] =4 and A[min] =A[2] = 15, A[2] > A[3] due 15 > 4 so it will change between of index j and min so index min = j = 3.
With j loop, index j will increase 1 unit so index j = 4, I will compare between element of j and element of min, due A[j] = A[4] =17 and A[min] =A[3] = 4, A[3] < A[4] due 4 < 17 so it won't change between of index j and min.
With j loop, index j will increase 1 unit so index j = 5, I will compare between element of j and element of min, due A[j] = A[5] =9 and A[min] =A[3] = 4, A[3] < A[5] due 4 < 9 so it won't change between of index j and min.
With j loop, index j will increase 1 unit so index j = 6, I will compare between element of j and element of min, due A[j] = A[6] =7 and A[min] =A[3] = 4, A[3] < A[6] due 4 < 7 so it won't change between of index j and min.
With j loop, index j will increase 1 unit so index j = 7, I will compare between element of j and element of min, due A[j] = A[7] =6 and A[min] =A[3] = 4, A[3] < A[7] due 4 < 6 so it won't change between of index j and min.

With j loop, index j will increase 1 unit so index j = 8, I will compare between element of j and element of min, due A[j] = A[8] =10 and A[min] =A[3] = 4, A[3] < A[8] due 4 < 10 so it won't change between of index j and min.

With j loop, index j will increase 1 unit so index j = 9, I will compare between element of j and element of min, due A[j] = A[9] =13 and A[min] =A[3] = 4, A[3] < A[9] due 4 < 13 so it won't change between of index j and min.

Due index j just browse until fewer than length of array A so j just run at j =9 after that, finish the j loop. I will swap between of the element of index min with element of i so swap A[min] = a[3] = 4 and A[i] = A[2] = 15 so array A will change below:

1	2	4	15	17	9	7	6	10	13

Step 4: with i =3, min =i then min = 3.

The first j loop, with j=i+1 = 3+1 = 4, I will compare between element of j and element of min, due A[j] = A[4] =17 and A[min] =A[3] = 15, A[3] < A[4] due 15 < 17 so it won't change between of index j and min.

With j loop, index j will increase 1 unit so index j = 5, I will compare between element of j and element of min, due A[j] = A[5] =9 and A[min] =A[3] = 15, A[3] > A[5] due 15 > 9 so it will change between of index j and min so index min = 5.

With j loop, index j will increase 1 unit so index j = 6, I will compare between element of j and element of min, due A[j] = A[6] =7 and A[min] =A[5] = 9, A[5] > A[6] due 9 > 7 so it will change between of index j and min so index min = 6.

With j loop, index j will increase 1 unit so index j = 7, I will compare between element of j and element of min, due A[j] = A[7] =6 and A[min] =A[6] = 7, A[6] > A[7] due 7 > 6 so it will change between of index j and min so index min = 7.

With j loop, index j will increase 1 unit so index j = 8, I will compare between element of j and element of min, due A[j] = A[8] =10 and A[min] =A[7] = 6, A[7] < A[8] due 6 < 10 so it won't change between of index j and min.

With j loop, index j will increase 1 unit so index j = 9, I will compare between element of j and element of min, due A[j] = A[9] =13 and A[min] =A[7] = 6, A[7] < A[9] due 6 < 13 so it won't change between of index j and min.

Due index j just browse until fewer than length of array A so j just run at j =9 after that, finish the j loop. I will swap between of the element of index min with element of i so swap A[min] = a[7] = 6 and A[i] = A[3] = 15 so array A will change below:

1	2	4	6	17	9	7	15	10	13

Step 5: with i =4, min =i then min = 4.

The first j loop, with j=i+1 = 4+1 = 5, I will compare between element of j and element of min, due A[j] = A[5] =9 and A[min] =A[4] = 17, A[4] > A[5] due 17 > 9 so it will change between of index j and min so index min = 5.

With j loop, index j will increase 1 unit so index j = 6, I will compare between element of j and element of min, due A[j] = A[6] =7 and A[min] =A[5] = 9, A[5] > A[6] due 9 > 7 so it will change between of index j and min so index min = 6.

With j loop, index j will increase 1 unit so index j = 7, I will compare between element of j and element of min, due A[j] = A[7] =15 and A[min] =A[6] = 7, A[6] < A[7] due 7 < 15 so it won't change between of index j and min.

With j loop, index j will increase 1 unit so index j = 8, I will compare between element of j and element of min, due A[j] = A[8] =10 and A[min] =A[6] = 7, A[6] < A[8] due 7 < 10 so it won't change between of index j and min.

With j loop, index j will increase 1 unit so index j = 9, I will compare between element of j and element of min, due A[j] = A[8] =13 and A[min] =A[6] = 7, A[6] < A[9] due 7 < 13 so it won't change between of index j and min.

Due index j just browse until fewer than length of array A so j just run at j =9 after that, finish the j loop. I will swap between of the element of index min with element of i so swap A[min] = a[6] = 7 and A[i] = A[4] = 17 so array A will change below:

1	2	4	6	7	9	17	15	10	13

Step 6: with i =5, min =i then min = 5.

The first j loop, with j=i+1 = 5+1 = 6, I will compare between element of j and element of min, due A[j] = A[6] =17 and A[min] =A[5] = 9, A[5] < A[6] due 9 < 17 so it won't change between of index j and min.

With j loop, index j will increase 1 unit so index j = 7, I will compare between element of j and element of min, due A[j] = A[7] =15 and A[min] =A[5] = 9, A[5] < A[7] due 9 < 15 so it won't change between of index j and min.

With j loop, index j will increase 1 unit so index j = 8, I will compare between element of j and element of min, due A[j] = A[8] =10 and A[min] =A[5] = 9, A[5] < A[8] due 9 < 10 so it won't change between of index j and min.

With j loop, index j will increase 1 unit so index j = 9, I will compare between element of j and element of min, due A[j] = A[9] =13 and A[min] =A[5] = 9, A[5] < A[9] due 9 < 13 so it won't change between of index j and min.

Due index j just browse until fewer than length of array A so j just run at j =9 after that, finish the j loop. I will swap between of the element of index min with element of i so swap A[min] = a[5] = 9 and A[i] = A[5] = 9 so array A will change below:

1	2	4	6	7	9	17	15	10	13

Step 7: with i =6, min =i then min = 6.
The first j loop, with j=i+1 = 6+1 = 7, I will compare between element of j and element of min, due A[j] = A[7] =15 and A[min] =A[6] = 17, A[6] > A[7] due 17 > 15 so it will change between of index j and min so index min = 7.

With j loop, index j will increase 1 unit so index j = 8, I will compare between element of j and element of min, due A[j] = A[8] =10 and A[min] =A[7] = 15, A[7] > A[8] due 15 > 10 so it will change between of index j and min so index min = 8.

With j loop, index j will increase 1 unit so index j = 9, I will compare between element of j and element of min, due A[j] = A[9] =13 and A[min] =A[8] = 10, A[8] < A[9] due 10 < 13 so it won't change between of index j and min.

Due index j just browse until fewer than length of array A so j just run at j =9 after that, finish the j loop. I will swap between of the element of index min with element of i so swap A[min] = a[8] = 10 and A[i] = A[6] = 17 so array A will change below:

1	2	4	6	7	9	10	15	17	13

Step 8: with i =7, min =i then min = 7.
The first j loop, with j=i+1 = 7+1 = 8, I will compare between element of j and element of min, due A[j] = A[8] =17 and A[min] =A[7] = 15, A[7] < A[8] due 15 < 17 so it won't change between of index j and min.

With j loop, index j will increase 1 unit so index j = 9, I will compare between element of j and element of min, due A[j] = A[9] =13 and A[min] =A[7] = 15, A[7] > A[9] due 15 > 13 so it will change between of index j and min so index min = 9.

Due index j just browse until fewer than length of array A so j just run at j =9 after that, finish the j loop. I will swap between of the element of index min with element of i so swap A[min] = a[9] = 13 and A[i] = A[7] = 15 so array A will change below:

1	2	4	6	7	9	10	13	17	15

Step 9: with i =8, min =i then min = 8.
The first j loop, with j=i+1 = 8+1 = 9, I will compare between element of j and element of min, due A[j] = A[9] = 15 and A[min] =A[8] = 17, A[8] > A[9] due 17 > 15 so it will change between of index j and min so index min = 9.

Due index j just browse until fewer than length of array A so j just run at j =9 after that, finish the j loop. I will swap between of the element of index min with element of i so swap A[min] = a[9] = 15 and A[i] = A[8] = 17 so array A will change below:

1	2	4	6	7	9	10	13	15	17

So index i just browse until fewer than the length of Array A minus -1 so i<9-1 so i<8 so the i loop will finish at here.And then, We had an array A which is sorted ascending below:

1	2	4	6	7	9	10	13	15	17

2.2 Theory b

The algorithm that I choose which is Quick Sort.

I have an array which is A[10]=[13,17,1,4,2,9,7,6,10,15], I will sort it to ascending.

Quick Sort is a divide algorithm, which take an element as pivot and partitions around this pivot. I will choose the last element as pivot. But the most important of this algorithm is partition() method, this method will take an element of array as pivot and I will compare this element of pivot to all elements before this pivot element if all elements before this pivot element will be smaller this pivot element ,then all smaller elements will put before this pivot element. After this, I will put this pivot element to right position in sorted array.

The below array A, it don't sorting, it original.

13	17	1	4	2	9	7	6	10	15

I have the low which is starting index and the high is ending index. And then, I will start from the leftmost element and following the index of smaller(or equal to) elements as i variable. And the high will be the index of last element in array. When browsing, If I find a smaller element, I will swap current element with element of index i. If not, I don't swap of them. Sometimes, the index i is the low minus 1.

With QuickSort() method, I will be passed three parameters including array A, low and high. With the low is starting the first index of array A and the high is ending the last index of array A. And I will create the pivot variable to calling the partition() method with three parameters including array A, low and high.

I will call QuickSort(A,0,9) method, due to low < high so I will create pivot variable is the result of partition(A,0,9) method I will calculate below:

> In partition(A,0,9) method, I will assign the element of index 9 to pivot variable which is A[9] = 15 and I will assign the index low minus 1 to i variable which is -1. I will use the for loop and browsing with index j from index low to index high and after each for loop I will plus 1 in index j. In the for loop, I will compare the element of j to pivot variable, if the pivot is larger the element of j, then I will plus 1 in i variable and swap between the the element of i with the element of j after this I will start the continue loop.If not, I will start the continue loop. After finishing the loop, I will swap the element of i plus 1 with the index high element of array. And then I will return index i plus 1, which is pivot variable of QuickSort() method.

The first loop, with j = low = 0, so A[j] = A[0] = 13 and compare with pivot variable, so A[j] < pivot because A[0] = 13 < pivot = 15 so i variable will plus 1 which is -1 + 1 = 0 and I swap A[j] with A[i] but index j will be equal with i variable.So I will continue the for loop.

The second loop, with j = 1, A[1] = 17 so A[1] > pivot because 17 > 15 so I will continue the for loop.

Continue loop, with j = 2, A[j] = A[2] = 1 so A[2] < pivot because 1 < 15 so i variable will plus 1 which is 0 + 1 = 1 and I swap A[j] = A[2] = 1 with A[i] =A[1] = 17,after this I will continue the for loop. So the array after swapping is below:

13	1	17	4	2	9	7	6	10	15

Continue loop, with j = 3, A[j] = A[3] = 4 so A[3] < pivot because 4 < 15 so i variable will plus 1 which is 1 + 1 = 2 and I swap A[j] = A[3] = 4 with A[i] =A[2] = 17,after this I will continue the for loop. So the array after swapping is below:

13	1	4	17	2	9	7	6	10	15

Continue loop, with j = 4, A[j] = A[4] = 2 so A[4] < pivot because 2 < 15 so i variable will plus 1 which is 2 + 1 = 3 and I swap A[j] = A[4] = 2 with A[i] =A[3] = 17,after this I will continue the for loop. So the array after swapping is below:

13	1	4	2	17	9	7	6	10	15

Continue loop, with j = 5, A[j] = A[5] = 9 so A[5] < pivot because 9 < 15 so i variable will plus 1 which is 3 + 1 = 4 and I swap A[j] = A[5] = 9 with A[i] =A[4] = 17,after this I will continue the for loop. So the array after swapping is below:

13	1	4	2	9	17	7	6	10	15

Continue loop, with j = 6, A[j] = A[6] = 7 so A[6] < pivot because 7 < 15 so i variable will plus 1 which is 4 + 1 = 5 and I swap A[j] = A[6] = 7 with A[i] =A[5] = 17,after this I will continue the for loop. So the array after swapping is below:

13	1	4	2	9	7	17	6	10	15

Continue loop, with j = 7, A[j] = A[7] = 6 so A[7] < pivot because 6< 15 so i variable will plus 1 which is 5 + 1 = 6 and I swap A[j] = A[7] = 6 with A[i] =A[6] = 17,after this I will continue the for loop. So the array after swapping is below:

13	1	4	2	9	7	6	17	10	15

Continue loop, with j = 8, A[j] = A[8] = 10 so A[8] < pivot because 10< 15 so i variable will plus 1 which is 6 + 1 = 7 and I swap A[j] = A[8] = 10 with A[i] =A[7] = 17, after this I will finish the for loop with index j = 8 because the for loop just browses smaller value 9 which is the high value. So the array after swapping is below:

13	1	4	2	9	7	6	10	17	15

After finishing the for loop, I have value i = 7. I will swap between of the element of i plus 1 and the element of value high so the i variable is 7+1 = 8 and high = 9 so swap A[i] = A[8] = 17 with A[high] = A[9] = 15 So the array after swapping is below:

13	1	4	2	9	7	6	10	15	17

After swapping of this, I will return value i plus 1 which is 9.

With the result of partition(A,0,9) method is 9 and it's still the value of pivot variable which is 9. So I will be recursion this QuickSort() method with three parameters like the first but with the high will change, it will browse until before pivot variable in QuickSort() method so It's recursion QuickSort(a,0,pivot -1)

method which is QuickSort(a,0,8) method, due to low < high so I will create pivot variable is the result of partition(A,0,8) method I will calculate below:

In partition(A,0,8) method, I will assign the element of index 8 to pivot variable which is A[8] = 15 and I will assign the index low minus 1 to i variable which is -1. I will use the for loop to browse and the progress browsing and the array is below:

13	1	4	2	9	7	6	10	15	17

The first loop, with j = low = 0, so A[j] = A[0] = 13 and compare with pivot variable, so A[j] < pivot because A[0] = 13 < pivot = 15 so i variable will plus 1 which is -1 + 1 = 0 and I swap A[j] with A[i] but index j will be equal with i variable.So I will continue the for loop.

The second loop, with j = 1, A[1] = 1 so A[1] < pivot because 1 < 15 so i variable will plus 1 which is 0 + 1 = 1 and I swap A[j] = A[1] = 1 with A[i] =A[1] = 1 but index j will be equal with i variable.So I will continue the for loop.

Continue loop, with j = 2, A[j] = A[2] = 4 so A[2] < pivot because 4 < 15 so i variable will plus 1 which is 1 + 1 = 2 and I swap A[j] = A[2] = 1 with A[i] =A[2] = 4 but index j will be equal with i variable.So I will continue the for loop.

Continue loop, with j = 3, A[j] = A[3] = 2 so A[3] < pivot because 2 < 15 so i variable will plus 1 which is 2 + 1 = 3 and I swap A[j] = A[3] = 2 with A[i] =A[3] = 2 but index j will be equal with the value of i variable.So I will continue the for loop.

Continue loop, with j = 4, A[j] = A[4] = 9 so A[4] < pivot because 9 < 15 so i variable will plus 1 which is 3 + 1 = 4 and I swap A[j] = A[4] = 9 with A[i] =A[4] = 9 but index j will be equal with the value of i variable.So I will continue the for loop.

Continue loop, with j = 5, A[j] = A[5] = 7 so A[5] < pivot because 7 < 15 so i variable will plus 1 which is 4 + 1 = 5 and I swap A[j] = A[5] = 7 with A[i] =A[5] = 7 but index j will be equal with the value of i variable.So I will continue the for loop.

Continue loop, with j = 6, A[j] = A[6] = 6 so A[6] < pivot because 6 < 15 so i variable will plus 1 which is 5 + 1 = 6 and I swap A[j] = A[6] = 6 with A[i] =A[6] = 6 but index j will be equal with the value of i variable.So I will continue the for loop.

Continue loop, with j = 7, A[j] = A[7] = 10 so A[7] < pivot because 10 < 15 so i variable will plus 1 which is 6 + 1 = 7 and I swap A[j] = A[7] = 10 with A[i] =A[7] = 10 but index j will be equal with the value of i variable. After this I will finish the for loop with index j = 7 because the for loop just browses smaller value 8 which is the high value. So the array after swapping is below:

13	1	4	2	9	7	6	10	15	17

After finishing the for loop, I have value i = 7. I will swap between of the element of i plus 1 and the element of value high so the i variable is 7+1 = 8 and value high = 8 so swap A[i] = A[8] = 15 with A[high] = A[8] = 15 but index j will be equal with the value of i variable so it swap is similar the value.

After swapping of this, I will return value i plus 1 which is 8.

With the result of partition(A,0,8) method is 8 and it's still the value of pivot variable which is 8. So I will be recursion this QuickSort() method with three parameters like the first but with the high will change, it will browse until before pivot variable in QuickSort() method so It's recursion QuickSort(a,0,pivot -1)

method which is QuickSort(a,0,7) method, due to low < high so I will create pivot variable is the result of partition(A,0,7) method I will calculate below:

In partition(A,0,7) method, I will assign the element of index 7 to pivot variable which is A[7] = 10 and I will assign the index low minus 1 to i variable which is -1. I will use the for loop to browse and the progress browsing and the array is below:

13	1	4	2	9	7	6	10	15	17

The first loop, with j = low = 0, so A[j] = A[0] = 13 and compare with pivot variable, so A[j] > pivot because A[0] = 13 > pivot = 10. So I will continue the for loop.

The second loop, with j = 1, A[1] = 1 so A[1] < pivot because 1 < 10 so the i variable will plus 1 which is -1 + 1 = 0 and I swap A[j] = A[1] = 1 with A[i] =A[0] = 13, after this I will continue the for loop. So the array after swapping is below:

1	13	4	2	9	7	6	10	15	17

Continue loop, with j = 2, A[j] = A[2] = 4 so A[2] < pivot because 4 < 10 so the i variable will plus 1 which is 0 + 1 = 1 and I swap A[j] = A[2] = 4 with A[i] =A[1] = 13, after this I will continue the for loop. So the array after swapping is below:

1	4	13	2	9	7	6	10	15	17

Continue loop, with j = 3, A[j] = A[3] = 2 so A[3] < pivot because 2 < 10 so the i variable will plus 1 which is 1 + 1 = 2 and I swap A[j] = A[3] = 2 with A[i] =A[2] = 13, after this I will continue the for loop. So the array after swapping is below:

1	4	2	13	9	7	6	10	15	17

Continue loop, with j = 4, A[j] = A[4] = 9 so A[4] < pivot because 9 < 10 so the i variable will plus 1 which is 2 + 1 = 3 and I swap A[j] = A[4] = 9 with A[i] =A[3] = 13, after this I will continue the for loop. So the array after swapping is below:

1	4	2	9	13	7	6	10	15	17

Continue loop, with j = 5, A[j] = A[5] = 7 so A[5] < pivot because 7 < 10 so the i variable will plus 1 which is 3 + 1 = 4 and I swap A[j] = A[5] = 7 with A[i] =A[4] = 13, after this I will continue the for loop. So the array after swapping is below:

1	4	2	9	7	13	6	10	15	17

Continue loop, with j = 6, A[j] = A[6] = 6 so A[6] < pivot because 6 < 10 so the i variable will plus 1 which is 4 + 1 = 5 and I swap A[j] = A[6] = 6 with A[i] =A[5] = 13. After this I will finish the for loop with index j = 6 because the for loop just browses smaller value 7 which is the high value. So the array after swapping is below:

1	4	2	9	7	6	13	10	15	17

After finishing the for loop, I have value i = 5. I will swap between of the element of i plus 1 and the element of value high so the i variable is 5+1 = 6 and value high = 7 so swap A[i] = A[6] = 13 with A[high] = A[7] = 10 so I will swap between of them. So the array after swapping is below:

1	4	2	9	7	6	10	13	15	17

After swapping of this, I will return value i plus 1 which is 5 + 1 = 6.

With the result of partition(A,0,7) method is 6 and it's still the value of pivot variable which is 6. So I will be recursion this QuickSort() method with three parameters like the first but with the high will change, it will browse until before pivot variable in QuickSort() method so It's recursion QuickSort(a,0,pivot -1)

method which is QuickSort(a,0,5) method, due to low < high so I will create pivot variable is the result of partition(A,0,5) method I will calculate below:

In partition(A,0,5) method, I will assign the element of index 5 to pivot variable which is A[5] = 6 and I will assign the index low minus 1 to i variable which is -1. I will use the for loop to browse and the progress browsing and the array is below:

1	4	2	9	7	6	10	13	15	17

The first loop, with j = low = 0, so A[j] = A[0] = 1 and compare with pivot variable, so A[j] < pivot because A[0] = 1 < pivot = 6 so the i variable will plus 1 which is -1 + 1 = 0 and I swap A[j] = A[0] = 1 with A[i] =A[0] = 1 but index j will be equal with the value of i variable.So I will continue the for loop.

The second loop, with j = 1, A[1] = 4 so A[1] < pivot because 4 < 6 so the i variable will plus 1 which is 0 + 1 = 1 and I swap A[j] = A[1] = 4 with A[i] =A[1] = 4 but index j will be equal with the value of i variable.So I will continue the for loop.

Continue loop, with j = 2, A[j] = A[2] = 2 so A[2] < pivot because 2 < 6 so the i variable will plus 1 which is 1 + 1 = 2 and I swap A[j] = A[2] = 2 with A[i] =A[2] = 2, but index j will be equal with the value of i variable.So I will continue the for loop.

Continue loop, with j = 3, A[j] = A[3] = 9 so A[3] > pivot because 9 > 6 So I will continue the for loop.

Continue loop, with j = 4, A[j] = A[4] = 7 so A[4] > pivot because 7 > 6. So I will finish the for loop with index j = 4 because the for loop just browses smaller value 5 which is the high value. So the array is below:

1	4	2	9	7	6	10	13	15	17

After finishing the for loop, I have value i = 2. I will swap between of the element of i plus 1 and the element of value high so the i variable is 2+1 = 3 and value high = 5 so swap A[i] = A[3] = 9 with A[high] = A[5] = 6 so I will swap between of them. So the array after swapping is below:

1	4	2	6	7	9	10	13	15	17

After swapping of this, I will return value i plus 1 which is 2 + 1 = 3.

With the result of partition(A,0,5) method is 3 and it's still the value of pivot variable which is 3. So I will be recursion this QuickSort() method with three parameters like the first but with the high will change, it will browse until before pivot variable in QuickSort() method so It's recursion QuickSort(a,0,pivot -1) method which is QuickSort(a,0,2) method, due to low < high so I will create pivot variable is the result of partition(A,0,2) method I will calculate below:

In partition(A,0,2) method, I will assign the element of index 2 to pivot variable which is A[2] = 2 and I will assign the index low minus 1 to i variable which is -1. I will use the for loop to browse and the progress browsing and the array is below:

1	4	2	6	7	9	10	13	15	17

The first loop, with j = low = 0, so A[j] = A[0] = 1 and compare with pivot variable, so A[j] < pivot because A[0] = 1 < pivot = 2 so the i variable will plus 1 which is -1 + 1 = 0 and I swap A[j] = A[0] = 1 with A[i] =A[0] = 1 but index j will be equal with the value of i variable.So I will continue the for loop.

The second loop, with j = 1, A[1] = 4 so A[1] > pivot because 4 < 2 So I will finish the for loop with index j = 1 because the for loop just browses smaller value 2 which is the high value. So the array is below:

1	4	2	6	7	9	10	13	15	17

After finishing the for loop, I have value i = 0. I will swap between of the element of i plus 1 and the element of value high so the i variable is 0+1 = 1 and value high = 2 so swap A[i] = A[1] = 4 with A[high] = A[2] = 2 so I will swap between of them. So the array after swapping is below:

1	2	4	6	7	9	10	13	15	17

After swapping of this, I will return value i plus 1 which is 0 + 1 = 1.

With the result of partition(A,0,2) method is 1 and it's still the value of pivot variable which is 1. So I will be recursion this QuickSort() method with three parameters like the first but with the high will change, it will browse until before pivot variable in QuickSort() method so It's recursion QuickSort(a,0,pivot -1) method which is QuickSort(a,0,0) method, due to low = high so it's not correct with if condition so it will finish QuickSort() method at here.

Finally, I have the array which is sorted below:

1	2	4	6	7	9	10	13	15	17

CHAPTER 3 – QUESTION 3: STACK

3.1 Theory Reverse Polish Notation

Reverse Polish Notation is a method that performances mathematical expressions including the operator and the operand.The operators is following the operands of them.The expression must to read from left to right and they using the last two numbers as the operands.In the brackets will be priority higher than the operators and after using this, the brackets need to remove in postfix notation.Reverse Polish Notation is like the infix notation after using the methods then the result is the Postfix notation.

Using stack to create reverse polish notation:

I will create a stack which name operators that contains the operators and a different stack which name is postfix that contains postfix notation.

The operators with precedence following ascending from “* / + - (“.

If meeting a number then I will add it on the postfix stack. Or meeting an operator, if the operators stack is empty then I will add it on the operators stack.Opposite,I will compare this operators with the last operators stack, if this operator hasn’t be more precedence than the last operators stack then I will add the last operators stack in last postfix stack and adding this operator in the last operators stack.Opposite, if this operator has be more precedence than the last operators stack then I will add it in the last operators stack that waiting to add on the postfix stack.Besides that, if meeting an open parenthesis then will add it on the operators stack.When meeting a close parenthesis so I will add all the operators stack in the last postfix stack and removing all the open parenthesis in the operators stack until the operators is empty.

For instance, I have an infix expression: (5+3)*5/(8-6)+(2*5) that is the infix notation. I will use the algorithm to transform the expression to postfix notation:

I will create a stack which name operators that contains the operators and a different stack which name is postfix that contains postfix notation.

I will browse the infix expression:

The first, I will see the open parenthesis so I will add it on the operators stack so the operators stack includes “(“.

Continue, I see number 5 because it’s a number so I will add it on the postfix stack which includes “5”.

Continue, I will see “+” as the operator.The operators stack is not empty so I will compare “+” with “(“ of the last operators stack, We can see the

"+" which has be precedence than "(" so I will add it on the last operators stack so the operators stack includes "(", "+".

Continue, I see number 3 because it's a number so I will add it on the last postfix stack which includes "5 3".

Continue, I will see a close parenthesis so I will add all the operators stack which is waiting to add unless the open parenthesis, just add the operators in this.So the operators stack includes "(", "+" which will add "+" in the last postfix stack. the postfix stack which includes "5 3 +". And then I have to remove "+" and an open parenthesis in the operators stack so the operators stack is empty.

Continue, I will see "*" as the operator. Because the operators stack is empty so I will add it on the operators stack so the operators stack includes "*".

Continue, I see number 5 because it's a number so I will add it on the last postfix stack which includes "5 3 + 5".

Continue, I will see "/" as the operator.The operators stack is not empty so I will compare "/" with "*" of the last operators stack, We can see the "*" which has be precedence than "/" so I will add "*" on the last postfix stack and removing "*" in the operators stack.At the same time, I will add "/" in the last operators stack so the operators stack includes "/". The postfix stack which includes "5 3 + 5 * " and the operators stack includes "/".

Continue, I will see the open parenthesis so I will add it on the last operators stack so the operators stack includes "/" ,"(".

Continue, I see number 8 because it's a number so I will add it on the last postfix stack which includes "5 3 + 5 * 8".

Continue, I will see “-” as the operator.The operators stack is not empty so I will compare “-” with “(“ of the last operators stack, We can see the “-” which has be precedence than “(“ so I will add “*” on the last operators stack so I will add it on the last operators stack so the operators stack includes “/” , “(“, “-”.

Continue, I see number 6 because it’s a number so I will add it on the last postfix stack which includes “5 3 + 5 * 8 6”.

Continue, I will see a close parenthesis (“)”) so I will add all the operators stack which is waiting to add unless the open parenthesis, just add the operators in this.So the operators stack includes “/” , “(“, “-” which will add “-” in the last postfix stack. the postfix stack which includes “5 3 + 5 * 8 6 -”. And then I have to remove “-” and an open parenthesis in the operators stack but the operators stack is not empty, it still has “/” so still continue to add bacause the operators stack is not empty, it add “/” in the last postfix stack which includes “5 3 + 5 * 8 6 - / “.After this, I have to remove “/” in the operators stack.The operators stack is empty.The postfix stack contains “5 3 + 5 * 8 6 - /”.

Continue, I will see “+” as the operator because the operators stack is empty so I will add it on the last operators stack so the operators stack includes “+”.

Continue, I will see the open parenthesis so I will add it on the last operators stack so the operators stack includes “+” ,“(“.

Continue, I see number 2 because it’s a number so I will add it on the last postfix stack which includes “5 3 + 5 * 8 6 - / 2”.

Continue, I will see “*” as the operator.Due “*” has be more precedence than “(” in the last operators stack so I will add “*” on the last operators stack so the operators stack includes “+”, “(“, “*”.

Continue, I see number 5 because it’s a number so I will add it on the last postfix stack which includes “5 3 + 5 * 8 6 - / 2 5”.

Continue, I will see a close parenthesis (“)”) so I will add all the operators stack which is waiting to add unless the open parenthesis, just add the operators in this.So the operators stack includes “+”, “(“, “*” which will add “*” in the last postfix stack. the postfix stack which includes “5 3 + 5 * 8 6 - / 2 5 *”. And then I have to remove “*” and an open parenthesis in the operators stack but the operators stack is not empty, it still has “+” so still continue to add bacause the operators stack is not empty, it add “+” in the last postfix stack which includes “5 3 + 5 * 8 6 - / 2 5 * +” .After this, I have to remove “+” in the operators stack.The operators stack is empty.The postfix stack contains “5 3 + 5 * 8 6 - / 2 5 * + “.

Finally, We have the postfix notation which is “5 3 + 5 * 8 6 - / 2 5 * +”.

3.2 Theory to calculate the result of postfix notation

After 3.1, so we have the postfix notation which is “5 3 + 5 * 8 6 - / 2 5 * +”.

I will create a stack which name is stackCalculate to calculate in a stack.If browse the postfix notation that meeting element which is a number then I will add in the last stackCalculate.Opposite,if meeting an operator, then I will take two element of the last stackCalculate to calculate with an operator and removing two element in the last stackCalculate. After that, I had the result of the operator between of two element so I will add this result in the last stackCalculate via syntax “stackCalculate .push()” and it repeats until the last element of postfix notation.

The first, I will create two variables with integer data type that contain two value of the last stackCalculate.Specific as is last1 and last2 and a stack which is stackCalculate.Starting browse postfix notation:

The first, I will meet a number which is number 5 because it is a number so I will add this in the last stackCalculate so the stackCalculate contains "5".

Continue, I will meet a number which is number 3 because it is a number so I will add this in the last stackCalculate so the stackCalculate contains "5", "3".

Continue, I will meet an operator which is "+" so I will take a last element of stackCalculate to assign in stack2 and remove this in stackCalculate so stack2 = 3.Besides that, I will take a last element of stackCalculate to assign in stack1 and remove this in stackCalculate so stack1 = 5. After that, I will calculate stack1 + stack2 = 5 +3 = 8 so I will adding this result in the last stackCalculate.So the stackCalculate contains "8".

Continue, I will meet a number which is number 5 because it is a number so I will add this in the last stackCalculate so the stackCalculate contains "8", "5".

Continue, I will meet an operator which is "*" so I will take a last element of stackCalculate to assign in stack2 and remove this in stackCalculate so stack2 = 5.Besides that, I will take a last element of stackCalculate to assign in stack1 and remove this in stackCalculate so stack1 = 8. After that, I will calculate stack1 * stack2 = 8 * 5 = 40 so I will adding this result in the last stackCalculate.So the stackCalculate contains "40".

Continue, I will meet a number which is number 8 because it is a number so I will add this in the last stackCalculate so the stackCalculate contains "40", "8".

Continue, I will meet a number which is number 6 because it is a number so I will add this in the last stackCalculate so the stackCalculate contains “40”, “8”, “6”.

Continue, I will meet an operator which is “-” so I will take a last element of stackCalculate to assign in stack2 and remove this in stackCalculate so stack2 = 6.Besides that, I will take a last element of stackCalculate to assign in stack1 and remove this in stackCalculate so stack1 = 8. After that, I will calculate stack1 * stack2 = 8 - 6 = 2 so I will adding this result in the last stackCalculate.So the stackCalculate contains “40”, “2”.

Continue, I will meet an operator which is “/” so I will take a last element of stackCalculate to assign in stack2 and remove this in stackCalculate so stack2 = 2.Besides that, I will take a last element of stackCalculate to assign in stack1 and remove this in stackCalculate so stack1 = 40. After that, I will calculate stack1 / stack2 = 40 / 2 = 20 so I will adding this result in the last stackCalculate.So the stackCalculate contains “20”.

Continue, I will meet a number which is number 2 because it is a number so I will add this in the last stackCalculate so the stackCalculate contains “20”, “2”.

Continue, I will meet a number which is number 5 because it is a number so I will add this in the last stackCalculate so the stackCalculate contains “20”, “2”, “5”.

Continue, I will meet an operator which is “*” so I will take a last element of stackCalculate to assign in stack2 and remove this in stackCalculate so stack2 = 5.Besides that, I will take a last element of stackCalculate to assign in stack1 and remove this in stackCalculate so stack1 = 2. After that, I will calculate stack1 * stack2 = 2 * 5 = 10 so I will adding this result in the last stackCalculate.So the stackCalculate contains “20”, “10”.

Continue, I will meet an operator which is “+” so I will take a last element of stackCalculate to assign in stack2 and remove this in stackCalculate so stack2 = 10.Besides that, I will take a last element of stackCalculate to assign in stack1 and remove this in stackCalculate so stack1 = 20. After that, I will calculate stack1 + stack2 = 20 + 10 = 30 so I will adding this result in the last stackCalculate.So the stackCalculate contains “30”. Due the “+” operator is the last element of the stackCalculate so I will finish browsing the postfix notaion and return the result.

Finally, I will take the element of the stackCalculate to return the result which is 30. So the result of postfix notation is 30.

CHAPTER 4 – QUESTION 4: DATA STRUTURES

4.1 Theory of data structures

The first of operations, add a new student:

With the problem of managing students using Array:

So the array is quite hard to add an element in array because it will use the for each on the first for.The strength of operation is

The strength of this data structure is just use the for loop and the algorithm is easy to understand.

The weakness of this data structure,it’s use two for loop and spending memory too much.

The efficiency of this data structure isn’t useful to add it so it isn’t applicable to remove a student because it’s quite long time and spending memory too much.

With the problem of managing students using Linked List:

The strength of this data structure is very easy understand and writing code to add a new student in linked list because it will assign a node which I want to add in this address to a new node student and then, a new node student will assign to the next this node which I want to add in this address.

The weakness of this data structure, we have to browse from the head node to null node so it wastes the memory of my computer and it's so long steam to add a new student.

The efficiency of this data structure is really convenient and useful.Because it's easy to code and understand the algorithm and spending few memories

With the problem of managing students using An AVL tree:

The strength of this data structure, it's so convenient because when I add a new student then an AVL tree so it will compare a new node student with a node root after comparing so it will add the left side if compare between of a new node student fewer than a node root. Opposite it will add in right side.After adding the side, it will balance with all node their side. Finally, the strength of this structure is so convenient to search and manage students and then we don't need to sort a tree because it's balanced.

The weakness of this data structure, it so hard to understand and write code because it's complicated and it wastes the memory and when running this structure so I saw it was slowly.

The efficiency of each data structure on add a new student:I think an AVL tree is efficiency because it don't need to sort a tree. When adding, it will balance at the same time. The rest of structures is quite convenient but it isn't as convenient as an AVL tree.

The efficiency of this data structure is really useful and convenient to add an new element because it can balance and the position of it will be placed at reasonable address.But it has few hard to code this.

The second of operations, remove a student:

With the problem of managing students using Array:

The strength of this data structure is so easy to understand it because just take a after element assigning to an element which before a removing element and do that until the last element but remember that must to minus 1 the length of array .

The weakness of this data structure, it's quite wasting time and memory. Because it uses two for loop to remove and browse a long time so I think it's the weakness.

The efficiency of this data structure isn't applicable to remove a student because it's quite long time so it's little convenient.

With the problem of managing students using Linked List:

The strength of this data structure isn't quite complex to remove any student, it just browses from node head to node null if meeting a node that want to remove then so just take a node before a node that removes and then assign to the next node that want to remove.It's so fast and wasting fewer the memory.

The weakness of this data structure, we have to create a the free memory to contains a node which deleted by assign to null.

The efficiency of this data structure, it’s so useful to remove a student because it’s easy to do that. Fast and succinct. I think this data structure is the most convenient to remove a student.

With the problem of managing students using An AVL tree:

The strength of this data structure, after removing a student so it will balance again the AVL tree to ensure balance of tree.

The weakness of this data structure, very complex to write code to remove and balance an AVL tree for this so it will spend some the memories for this code.

The efficiency of this data structure,it’s still quite convenient but it’s complex so The efficiency is quite convenient.

The third of operations, search a student:

With the problem of managing students using Array:

The strength of this data structure is quite easy to operation and understand, we just browse all elements of array and search a student that is quick.

The weakness of this data structure, it will spend some memories to use the loop and we have to browse all elements to look for a student.

The efficiency of this data structure, it’s so useful and cozy because the algorithm is so short to search a student.

With the problem of managing students using Linked List:

The strength of this data structure is so fluent to search a student, just need a while loop to browse all elements from head node to null. It’s similar an array.

The weakness of this data structure, it will spend some memories to use a while loop and we have to browse all elements from head node to null looking for a student.

The efficiency of this data structure is really efficient like an array and the time run it so short.

With the problem of managing students using An AVL tree:

The strength of this data structure,it can browse any side in AVL tree because it has many algorithms to browse an AVL tree as NodeLeftRight,RightNodeLeft,… But it often uses NodeLeftRight so it browses being very detail every node in AVL tree.

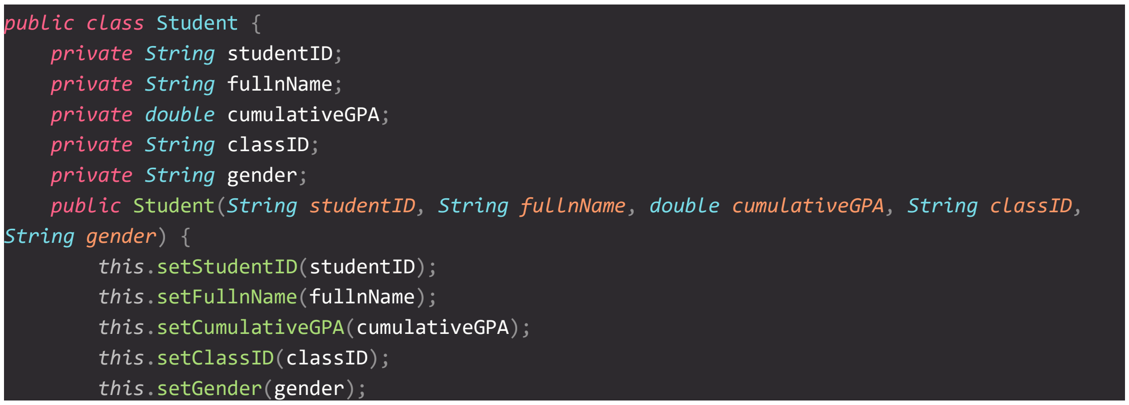
The weakness of this data structure, it spends the memory and time for search a student so it's quite complex.Besides that, it's hard to code to search this.

The efficiency of this data structure, it's a little convenient.As spending time and memory,few dreaming to code this.

4.2 Practical of data structures

In file Student.java, I created two properties which are classID and gender of student.

Student.java is below:



```java
public class Student {
    private String studentID;
    private String fullnName;
    private double cumulativeGPA;
    private String classID;
    private String gender;
    public Student(String studentID, String fullnName, double cumulativeGPA, String classID,
String gender) {
        this.setStudentID(studentID);
        this.setFullnName(fullnName);
        this.setCumulativeGPA(cumulativeGPA);
        this.setClassID(classID);
        this.setGender(gender);
    }
    public Student(){
        this.setStudentID("");
        this.setFullnName("");
        this.setCumulativeGPA(0.0);
        this.setClassID("");
        this.setGender("");
    }
    public String getGender() {
        return gender;
    }
    public void setGender(String gender) {
        this.gender = gender;
    }
    public String getClassID() {
        return classID;
    }
    public void setClassID(String classID) {
        this.classID = classID;
    }
    public double getCumulativeGPA() {
        return cumulativeGPA;
    }
    public void setCumulativeGPA(double cumulativeGPA) {
        this.cumulativeGPA = cumulativeGPA;
    }
    public String getFullnName() {
        return fullnName;
    }
    public void setFullnName(String fullnName) {
        this.fullnName = fullnName;
    }
    public String getStudentID() {
        return studentID;
    }
    public void setStudentID(String studentID) {
        this.studentID = studentID;
    }
    @Override
    public String toString() {
        return "student[" + getStudentID()  + "," + getFullnName() + "," + getCumulativeGPA() +
"," + getClassID() + "," + getGender()+"]";
    }
}
```

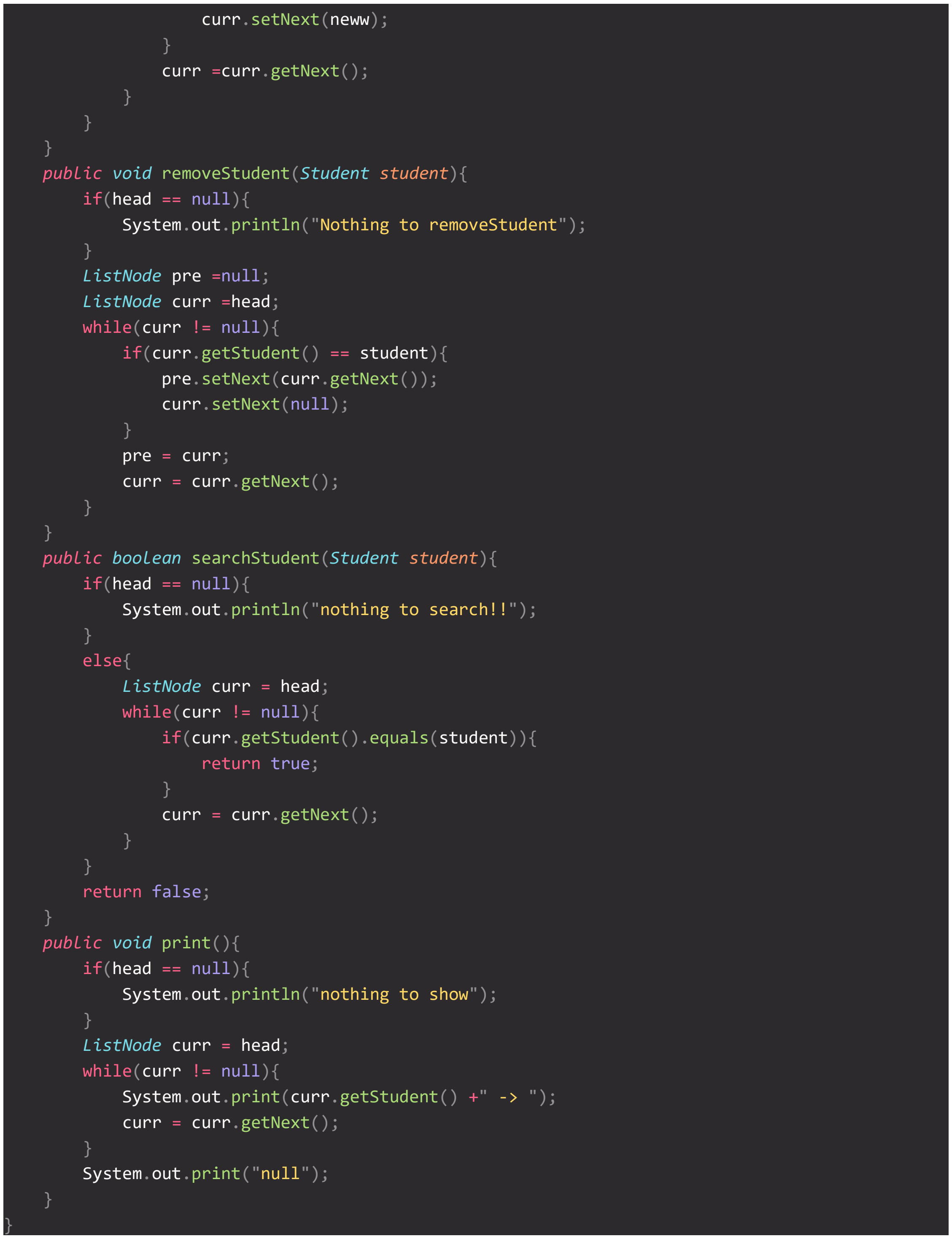
4.2.1 LinkedList

ListNode.java is below:

```java
public class ListNode{
    private Student student;
    private ListNode next;
    public ListNode(Student student, ListNode next) {
        this.setStudent(student);
        this.setNext(next);
    }
    public ListNode(Student student){
        this.setStudent(student);
        this.setNext(null);
    }
    public Student getStudent() {
        return student;
    }
    public void setStudent(Student student) {
        this.student = student;
    }
    public ListNode getNext() {
        return next;
    }
    public void setNext(ListNode next) {
        this.next = next;
    }
}
```

studentLinkedList.java (add a new student, remove a student and search a student) is below:

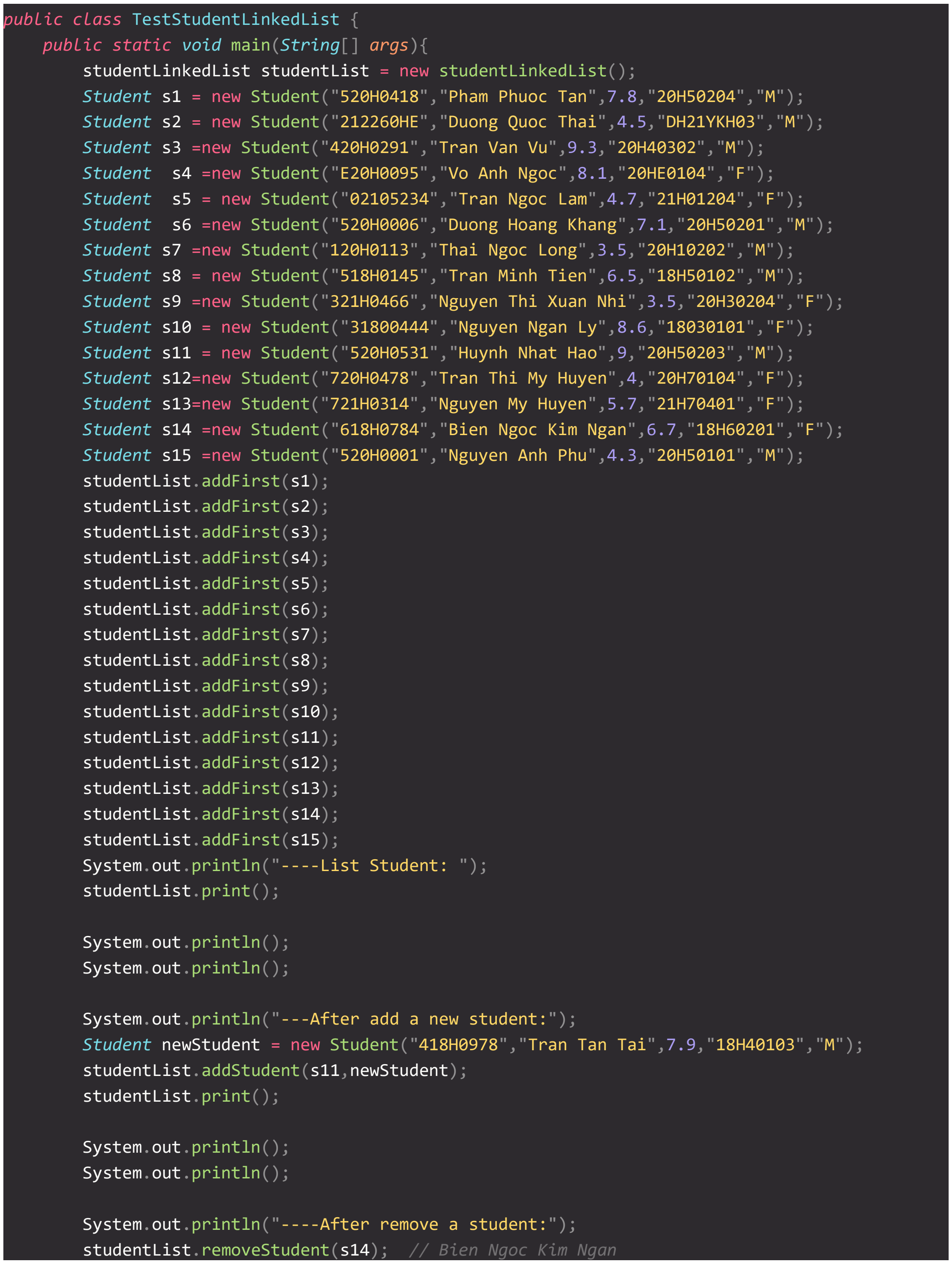
```java
public class studentLinkedList {
    private ListNode head;
    public studentLinkedList(){
        head =null;
    }
    public void addFirst(Student s){
        ListNode ne = new ListNode(s);
        ne.setNext(head);
        head = ne;
    }
    public void addStudent(Student studentKey,Student student){
        if(head == null){
            ListNode neww = new ListNode(student);
        }
        else{
            ListNode curr = head;
            while(curr != null){
                if(curr.getStudent() == studentKey){
                    ListNode neww = new ListNode(student,curr.getNext());
```



```java
                    curr.setNext(neww);
                }
                curr =curr.getNext();
            }
        }
    }
    public void removeStudent(Student student){
        if(head == null){
            System.out.println("Nothing to removeStudent");
        }
        ListNode pre =null;
        ListNode curr =head;
        while(curr != null){
            if(curr.getStudent() == student){
                pre.setNext(curr.getNext());
                curr.setNext(null);
            }
            pre = curr;
            curr = curr.getNext();
        }
    }
    public boolean searchStudent(Student student){
        if(head == null){
            System.out.println("nothing to search!!");
        }
        else{
            ListNode curr = head;
            while(curr != null){
                if(curr.getStudent().equals(student)){
                    return true;
                }
                curr = curr.getNext();
            }
        }
        return false;
    }
    public void print(){
        if(head == null){
            System.out.println("nothing to show");
        }
        ListNode curr = head;
        while(curr != null){
            System.out.print(curr.getStudent() +" -> ");
            curr = curr.getNext();
        }
        System.out.print("null");
    }
}
```

After this, I will test it.

TestStudentLinkedList.java is below:



```java
public class TestStudentLinkedList {
    public static void main(String[] args){
        studentLinkedList studentList = new studentLinkedList();
        Student s1 = new Student("520H0418","Pham Phuoc Tan",7.8,"20H50204","M");
        Student s2 = new Student("212260HE","Duong Quoc Thai",4.5,"DH21YKH03","M");
        Student s3 =new Student("420H0291","Tran Van Vu",9.3,"20H40302","M");
        Student  s4 =new Student("E20H0095","Vo Anh Ngoc",8.1,"20HE0104","F");
        Student  s5 = new Student("02105234","Tran Ngoc Lam",4.7,"21H01204","F");
        Student  s6 =new Student("520H0006","Duong Hoang Khang",7.1,"20H50201","M");
        Student s7 =new Student("120H0113","Thai Ngoc Long",3.5,"20H10202","M");
        Student s8 = new Student("518H0145","Tran Minh Tien",6.5,"18H50102","M");
        Student s9 =new Student("321H0466","Nguyen Thi Xuan Nhi",3.5,"20H30204","F");
        Student s10 = new Student("31800444","Nguyen Ngan Ly",8.6,"18030101","F");
        Student s11 = new Student("520H0531","Huynh Nhat Hao",9,"20H50203","M");
        Student s12=new Student("720H0478","Tran Thi My Huyen",4,"20H70104","F");
        Student s13=new Student("721H0314","Nguyen My Huyen",5.7,"21H70401","F");
        Student s14 =new Student("618H0784","Bien Ngoc Kim Ngan",6.7,"18H60201","F");
        Student s15 =new Student("520H0001","Nguyen Anh Phu",4.3,"20H50101","M");
        studentList.addFirst(s1);
        studentList.addFirst(s2);
        studentList.addFirst(s3);
        studentList.addFirst(s4);
        studentList.addFirst(s5);
        studentList.addFirst(s6);
        studentList.addFirst(s7);
        studentList.addFirst(s8);
        studentList.addFirst(s9);
        studentList.addFirst(s10);
        studentList.addFirst(s11);
        studentList.addFirst(s12);
        studentList.addFirst(s13);
        studentList.addFirst(s14);
        studentList.addFirst(s15);
        System.out.println("----List Student: ");
        studentList.print();

        System.out.println();
        System.out.println();

        System.out.println("---After add a new student:");
        Student newStudent = new Student("418H0978","Tran Tan Tai",7.9,"18H40103","M");
        studentList.addStudent(s11,newStudent);
        studentList.print();

        System.out.println();
        System.out.println();

        System.out.println("----After remove a student:");
        studentList.removeStudent(s14);  // Bien Ngoc Kim Ngan
        studentList.print();

        System.out.println();
        System.out.println();

        // Student searchStudent = new Student("520H0537","Tran Thi Hop",8.4,"20H50203","F");
        Student searchStudent = s1; // Pham phuoc Tan
        System.out.print("----Search a student: " + studentList.searchStudent(searchStudent));
    }
}
```

4.2.2 AVL tree

AVLNode.java is below:

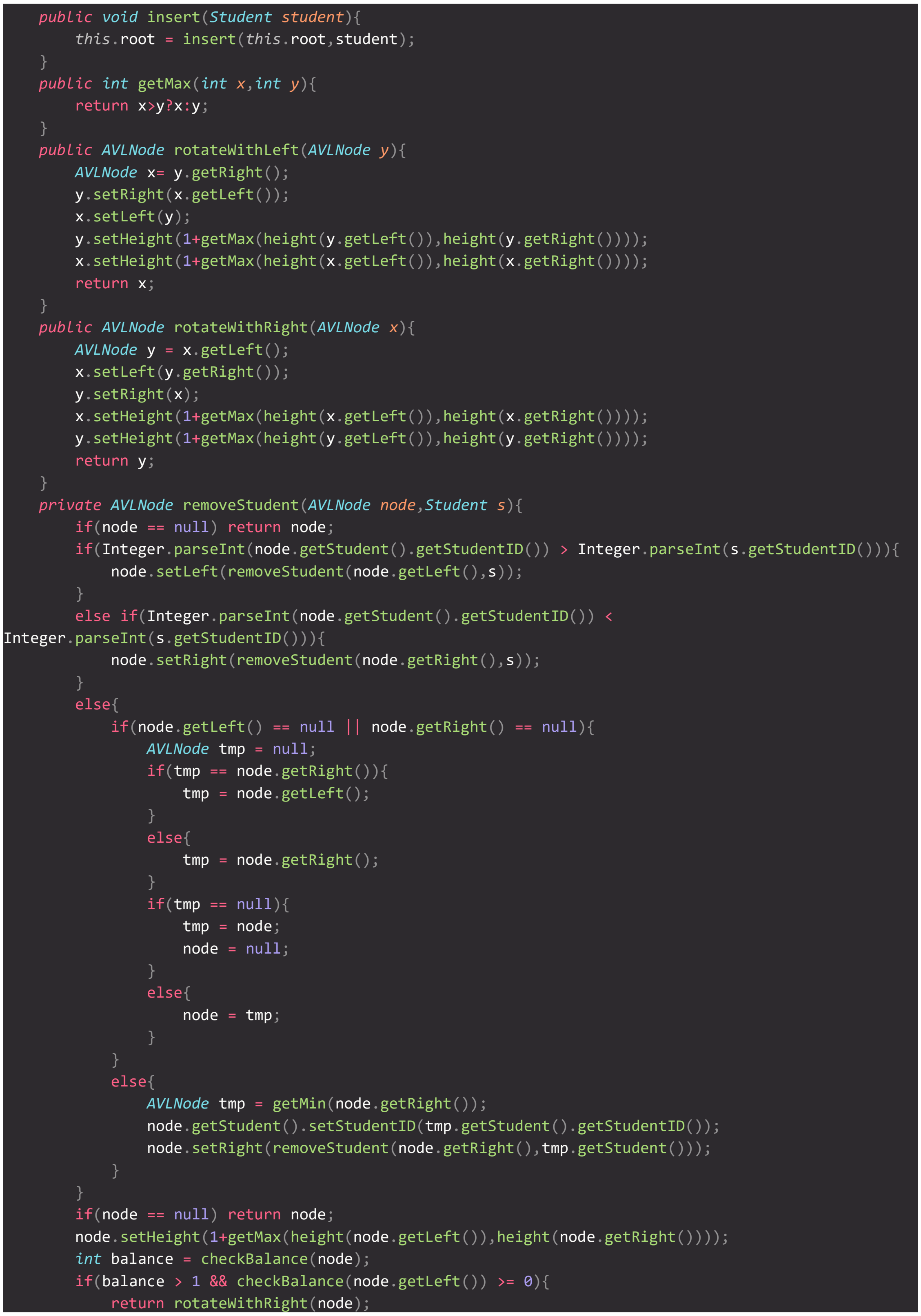
```java
public class AVLNode {
    private Student student;
    private AVLNode left,right;
    private int height;
    public AVLNode(Student student) {
        this.student = student;
        this.left = this.right = null;
        this.height = 0;
    }
    public Student getStudent() {
        return student;
    }
    public void setStudent(Student student) {
        this.student = student;
    }
    public AVLNode getRight() {
        return right;
    }
    public void setRight(AVLNode right) {
        this.right = right;
    }
    public AVLNode getLeft() {
        return left;
    }
    public void setLeft(AVLNode left) {
        this.left = left;
    }
    public int getHeight() {
        return height;
    }
    public void setHeight(int height) {
        this.height = height;
    }
}
```

AVLTree.java is below:

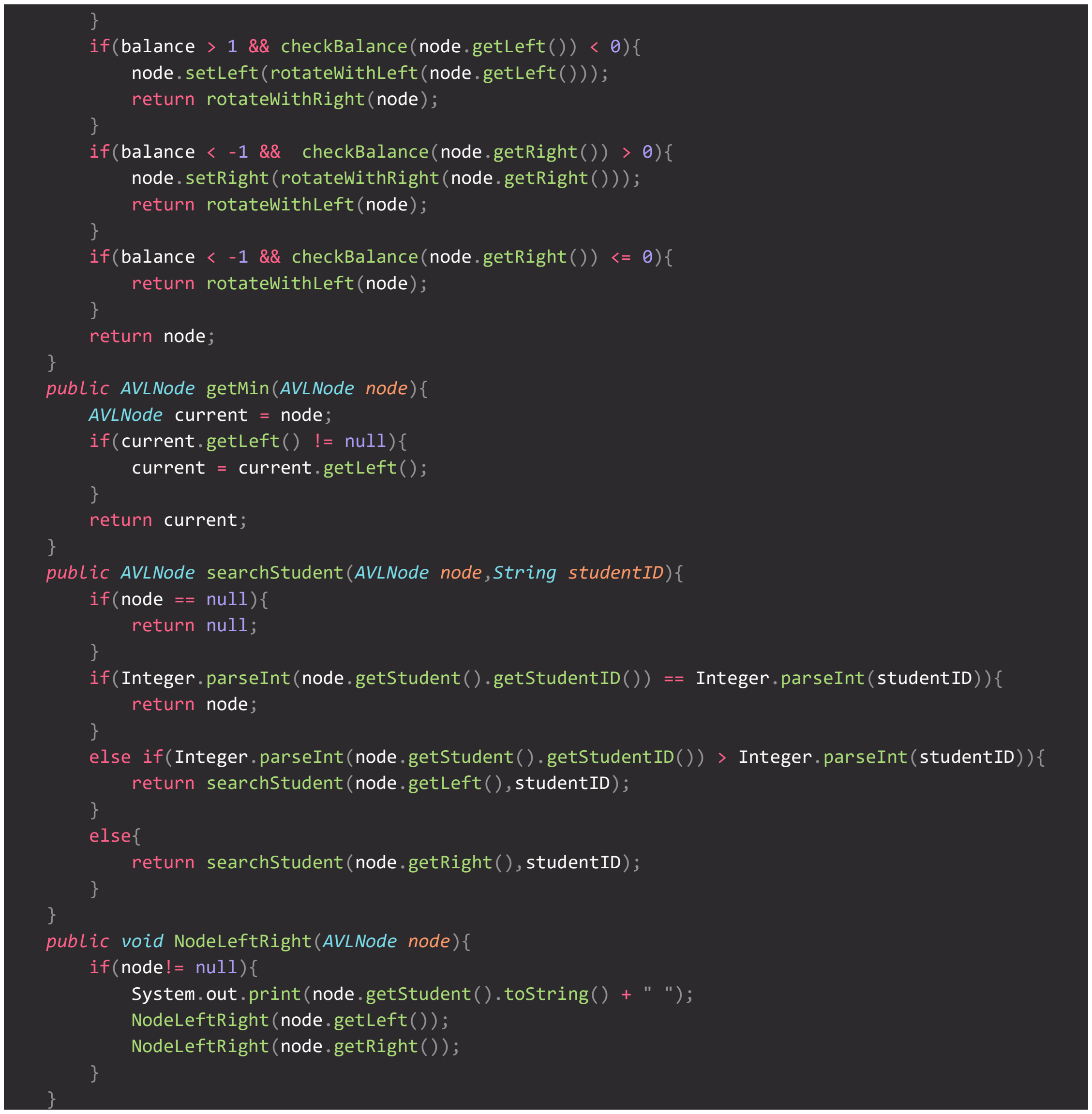
```java
public class AVLTree {
    private AVLNode root;

    public AVLTree() {
        this.setRoot(null);
    }
```

```java
    public AVLNode getRoot() {
        return root;
    }
```

```java
    public void setRoot(AVLNode root) {
        this.root = root;
    }
    public int height(AVLNode node){
        if(node == null) return 0;
        return node.getHeight();
    }
    public int checkBalance(AVLNode node){
        return height(node.getLeft()) - height(node.getRight());
    }
    private AVLNode insert(AVLNode node,Student student){
        if(node == null) return new AVLNode(student);
        if(Integer.parseInt(student.getStudentID()) <
Integer.parseInt(node.getStudent().getStudentID())){
            node.setLeft(insert(node.getLeft(), student));
        }
        else if(Integer.parseInt(student.getStudentID()) >
Integer.parseInt(node.getStudent().getStudentID())){
            node.setRight(insert(node.getRight(), student));
        }
        else{
            return node;
        }
        node.setHeight(1+getMax(height(node.getLeft()),height(node.getRight())));
        int balance = checkBalance(node);
        if(balance > 1 && Integer.parseInt(student.getStudentID()) <
Integer.parseInt(node.getLeft().getStudent().getStudentID())){
            return rotateWithRight(node);
        }
        if(balance > 1 && Integer.parseInt(student.getStudentID()) >
Integer.parseInt(node.getLeft().getStudent().getStudentID())){
            node.setLeft(rotateWithLeft(node.getLeft()));
            return rotateWithRight(node);
        }
        if(balance < -1 && Integer.parseInt(student.getStudentID()) <
Integer.parseInt(node.getLeft().getStudent().getStudentID())){
            node.setRight(rotateWithRight(node.getRight()));
            return rotateWithLeft(node);
        }
        if(balance < -1 && Integer.parseInt(student.getStudentID()) >
Integer.parseInt(node.getLeft().getStudent().getStudentID())){
            return rotateWithLeft(node);
        }
        return node;
    }
```

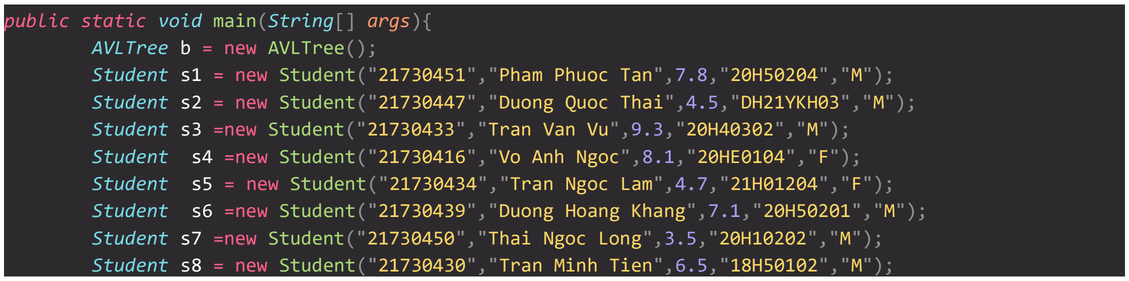


```
    public void insert(Student student){
        this.root = insert(this.root,student);
    }
    public int getMax(int x,int y){
        return x>y?x:y;
    }
    public AVLNode rotateWithLeft(AVLNode y){
        AVLNode x= y.getRight();
        y.setRight(x.getLeft());
        x.setLeft(y);
        y.setHeight(1+getMax(height(y.getLeft()),height(y.getRight())));
        x.setHeight(1+getMax(height(x.getLeft()),height(x.getRight())));
        return x;
    }
    public AVLNode rotateWithRight(AVLNode x){
        AVLNode y = x.getLeft();
        x.setLeft(y.getRight());
        y.setRight(x);
        x.setHeight(1+getMax(height(x.getLeft()),height(x.getRight())));
        y.setHeight(1+getMax(height(y.getLeft()),height(y.getRight())));
        return y;
    }
    private AVLNode removeStudent(AVLNode node,Student s){
        if(node == null) return node;
        if(Integer.parseInt(node.getStudent().getStudentID()) > Integer.parseInt(s.getStudentID())){
            node.setLeft(removeStudent(node.getLeft(),s));
        }
        else if(Integer.parseInt(node.getStudent().getStudentID()) <
Integer.parseInt(s.getStudentID())){
            node.setRight(removeStudent(node.getRight(),s));
        }
        else{
            if(node.getLeft() == null || node.getRight() == null){
                AVLNode tmp = null;
                if(tmp == node.getRight()){
                    tmp = node.getLeft();
                }
                else{
                    tmp = node.getRight();
                }
                if(tmp == null){
                    tmp = node;
                    node = null;
                }
                else{
                    node = tmp;
                }
            }
            else{
                AVLNode tmp = getMin(node.getRight());
                node.getStudent().setStudentID(tmp.getStudent().getStudentID());
                node.setRight(removeStudent(node.getRight(),tmp.getStudent()));
            }
        }
        if(node == null) return node;
        node.setHeight(1+getMax(height(node.getLeft()),height(node.getRight())));
        int balance = checkBalance(node);
        if(balance > 1 && checkBalance(node.getLeft()) >= 0){
            return rotateWithRight(node);
```

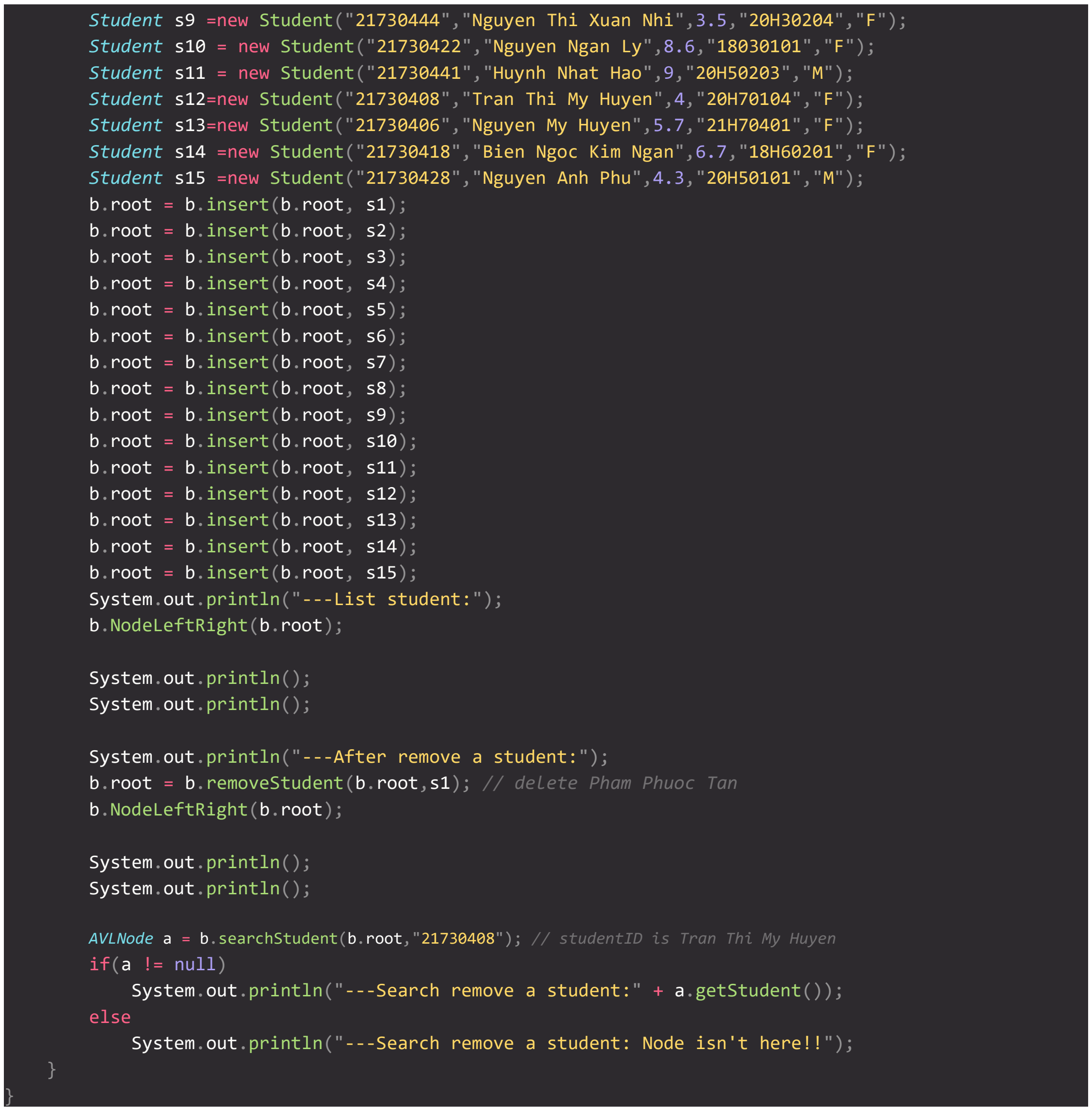


```
        }
        if(balance > 1 && checkBalance(node.getLeft()) < 0){
            node.setLeft(rotateWithLeft(node.getLeft()));
            return rotateWithRight(node);
        }
        if(balance < -1 &&  checkBalance(node.getRight()) > 0){
            node.setRight(rotateWithRight(node.getRight()));
            return rotateWithLeft(node);
        }
        if(balance < -1 && checkBalance(node.getRight()) <= 0){
            return rotateWithLeft(node);
        }
        return node;
    }
    public AVLNode getMin(AVLNode node){
        AVLNode current = node;
        if(current.getLeft() != null){
            current = current.getLeft();
        }
        return current;
    }
    public AVLNode searchStudent(AVLNode node,String studentID){
        if(node == null){
            return null;
        }
        if(Integer.parseInt(node.getStudent().getStudentID()) == Integer.parseInt(studentID)){
            return node;
        }
        else if(Integer.parseInt(node.getStudent().getStudentID()) > Integer.parseInt(studentID)){
            return searchStudent(node.getLeft(),studentID);
        }
        else{
            return searchStudent(node.getRight(),studentID);
        }
    }
    public void NodeLeftRight(AVLNode node){
        if(node!= null){
            System.out.print(node.getStudent().toString() + " ");
            NodeLeftRight(node.getLeft());
            NodeLeftRight(node.getRight());
        }
    }
```

Method main is to test AVL tree with add a new student, remove and search a student is below:



```
public static void main(String[] args){
        AVLTree b = new AVLTree();
        Student s1 = new Student("21730451","Pham Phuoc Tan",7.8,"20H50204","M");
        Student s2 = new Student("21730447","Duong Quoc Thai",4.5,"DH21YKH03","M");
        Student s3 =new Student("21730433","Tran Van Vu",9.3,"20H40302","M");
        Student  s4 =new Student("21730416","Vo Anh Ngoc",8.1,"20HE0104","F");
        Student  s5 = new Student("21730434","Tran Ngoc Lam",4.7,"21H01204","F");
        Student  s6 =new Student("21730439","Duong Hoang Khang",7.1,"20H50201","M");
        Student s7 =new Student("21730450","Thai Ngoc Long",3.5,"20H10202","M");
        Student s8 = new Student("21730430","Tran Minh Tien",6.5,"18H50102","M");
```



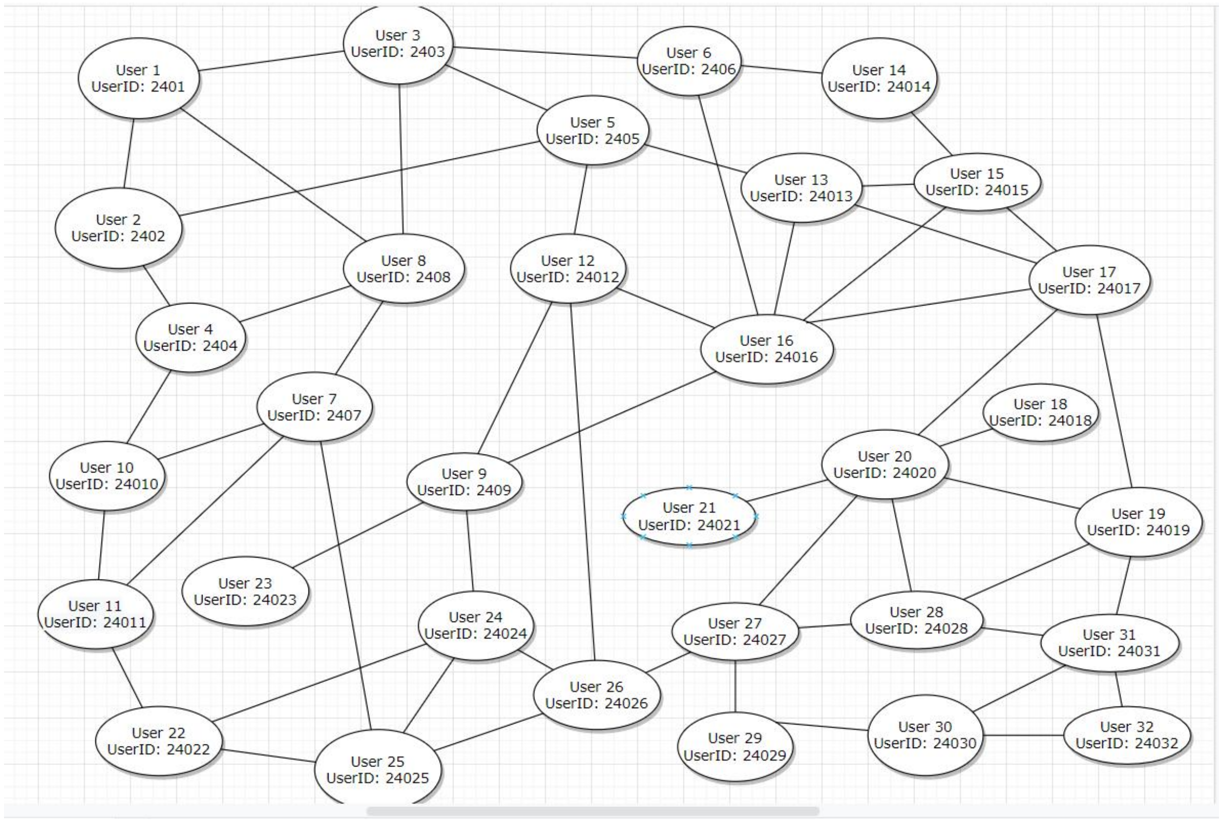
```
        Student s9 =new Student("21730444","Nguyen Thi Xuan Nhi",3.5,"20H30204","F");
        Student s10 = new Student("21730422","Nguyen Ngan Ly",8.6,"18030101","F");
        Student s11 = new Student("21730441","Huynh Nhat Hao",9,"20H50203","M");
        Student s12=new Student("21730408","Tran Thi My Huyen",4,"20H70104","F");
        Student s13=new Student("21730406","Nguyen My Huyen",5.7,"21H70401","F");
        Student s14 =new Student("21730418","Bien Ngoc Kim Ngan",6.7,"18H60201","F");
        Student s15 =new Student("21730428","Nguyen Anh Phu",4.3,"20H50101","M");
        b.root = b.insert(b.root, s1);
        b.root = b.insert(b.root, s2);
        b.root = b.insert(b.root, s3);
        b.root = b.insert(b.root, s4);
        b.root = b.insert(b.root, s5);
        b.root = b.insert(b.root, s6);
        b.root = b.insert(b.root, s7);
        b.root = b.insert(b.root, s8);
        b.root = b.insert(b.root, s9);
        b.root = b.insert(b.root, s10);
        b.root = b.insert(b.root, s11);
        b.root = b.insert(b.root, s12);
        b.root = b.insert(b.root, s13);
        b.root = b.insert(b.root, s14);
        b.root = b.insert(b.root, s15);
        System.out.println("---List student:");
        b.NodeLeftRight(b.root);

        System.out.println();
        System.out.println();

        System.out.println("---After remove a student:");
        b.root = b.removeStudent(b.root,s1); // delete Pham Phuoc Tan
        b.NodeLeftRight(b.root);

        System.out.println();
        System.out.println();

        AVLNode a = b.searchStudent(b.root,"21730408"); // studentID is Tran Thi My Huyen
        if(a != null)
            System.out.println("---Search remove a student:" + a.getStudent());
        else
            System.out.println("---Search remove a student: Node isn't here!!");
    }
}
```

CHAPTER 5 – QUESTION 5: GRAPH

5.1 Theory of graph

5.1.1 Theory of a

I have 32 vertices with userID and 55 edges:



Picture 5.1.1 Graph of User with each vertex which is userID

5.1.2 Theory of b

The result of traversing the graph using BFS starting from User 1 vertex:

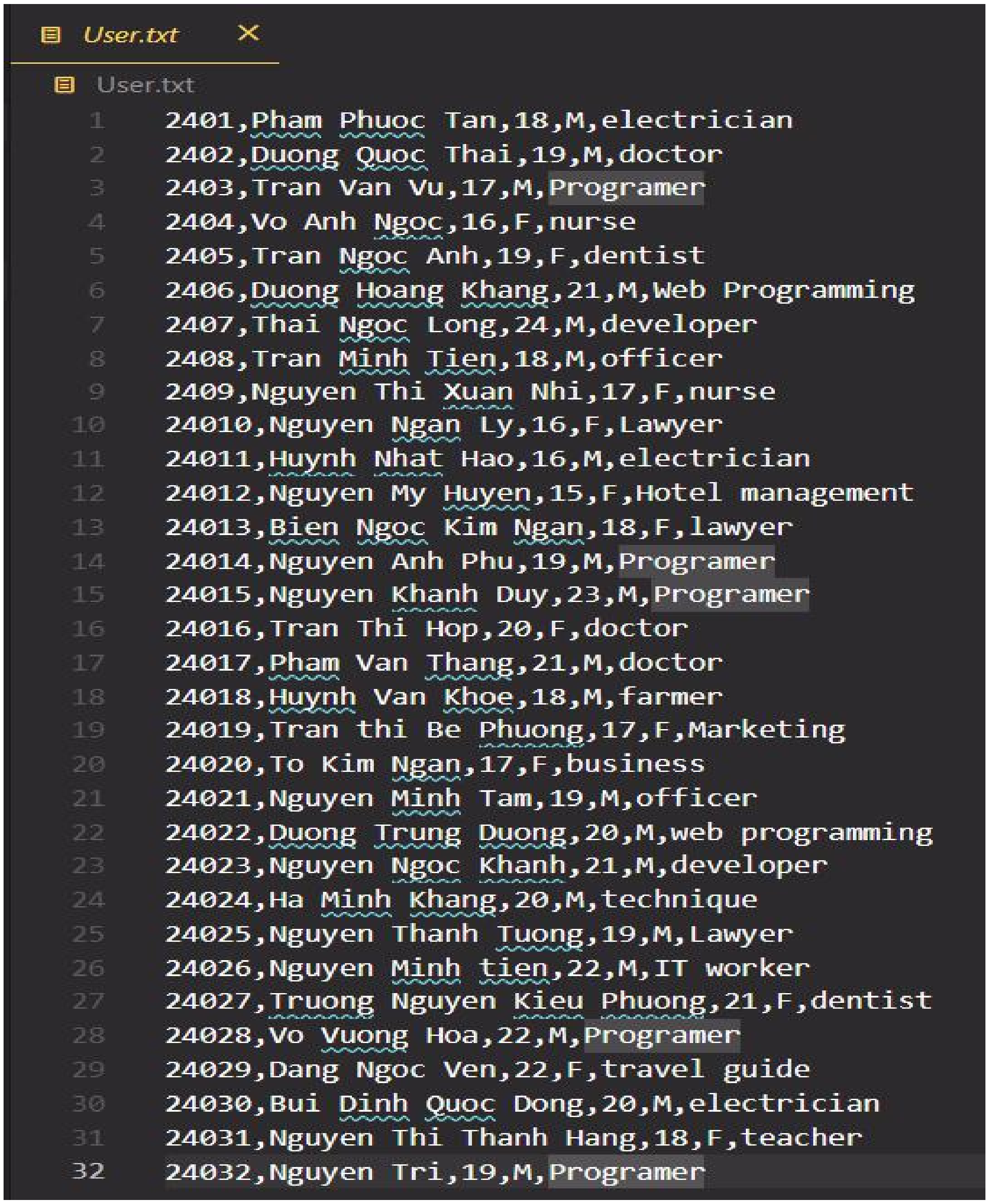
User1[UserID:2401] -> User2[UserID:2402] -> User3[UserID:2403] -> User8[UserID:2408] -> User4[UserID:2404] -> User5[UserID:2405] -> User6[UserID:2406] -> User7[UserID:2407] -> User10[UserID:24010] -> User12[UserID:24012] -> User13[UserID:24013] -> User14[UserID:24014] -> User16[UserID:24016] -> User11[UserID:24011] -> User25[UserID:24025] -> User9[UserID:2409] -> User26[UserID:24026] -> User15[UserID:24015] ->

User17[UserID:24017] -> User22[UserID:24022] -> User24[UserID:24024] -> User23[UserID:24023] -> User27[UserID:24027] -> User19[UserID:24019] -> User20[UserID:24020] -> User28[UserID:24028] -> User29[UserID:24029] -> User31[UserID:24031] -> User18[UserID:24018] -> User21[UserID:24021] -> User30[UserID:24030] -> User32[UserID:24032] .

The result of traversing the graph using DFS starting from User 1 vertex:
User1[UserID:2401] -> User2[UserID:2402] -> User4[UserID:2404] -> User8[UserID:2408] -> User3[UserID:2403] -> User5[UserID:2405] -> User12[UserID:24012] -> User9[UserID:2409] -> User16[UserID:24016] -> User6[UserID:2406] -> User14[UserID:24014] -> User15[UserID:24015] -> User13[UserID:24013] -> User17[UserID:24017] -> User19[UserID:24019] -> User20[UserID:24020] -> User18[UserID:24018] -> User21[UserID:24021] -> User27[UserID:24027] -> User26[UserID:24026] -> User24[UserID:24024] -> User22[UserID:24022] -> User11[UserID:24011] -> User7[UserID:2407] -> User10[UserID:24010] -> User25[UserID:24025] -> User28[UserID:24028] -> User31[UserID:24031] -> User30[UserID:24030] -> User29[UserID:24029] -> User32[UserID:24032] -> User23[UserID:24023] .

5.2 Practical of graph

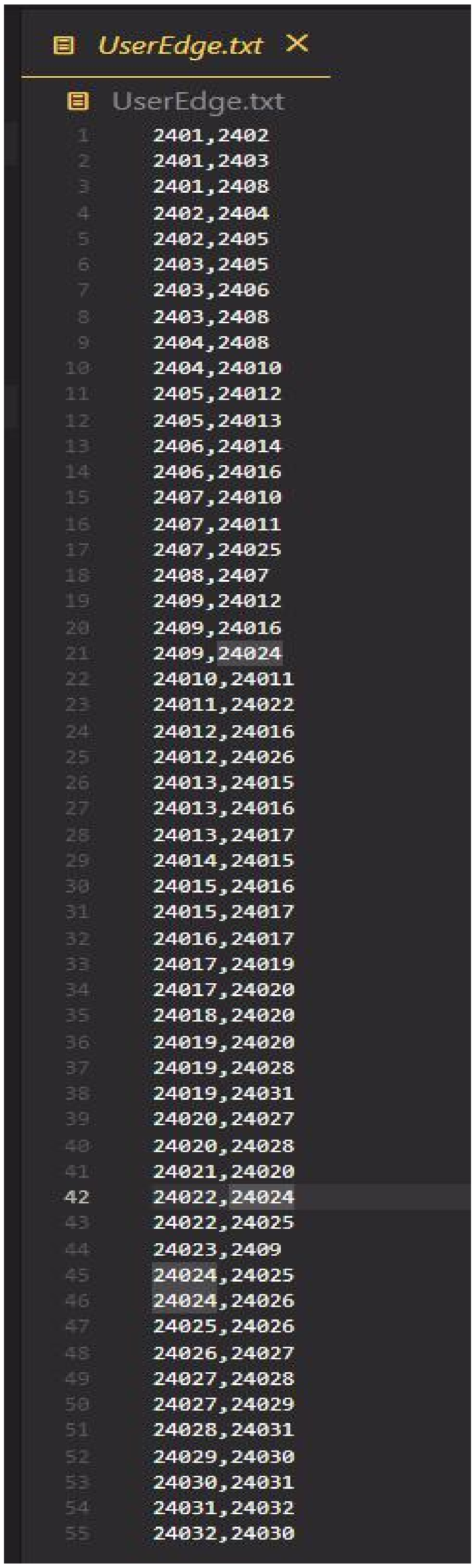
I create two file text including User.txt and UserEdge.txt which in below:



User.txt

```
2401,Pham Phuoc Tan,18,M,electrician
2402,Duong Quoc Thai,19,M,doctor
2403,Tran Van Vu,17,M,Programer
2404,Vo Anh Ngoc,16,F,nurse
2405,Tran Ngoc Anh,19,F,dentist
2406,Duong Hoang Khang,21,M,Web Programming
2407,Thai Ngoc Long,24,M,developer
2408,Tran Minh Tien,18,M,officer
2409,Nguyen Thi Xuan Nhi,17,F,nurse
24010,Nguyen Ngan Ly,16,F,Lawyer
24011,Huynh Nhat Hao,16,M,electrician
24012,Nguyen My Huyen,15,F,Hotel management
24013,Bien Ngoc Kim Ngan,18,F,lawyer
24014,Nguyen Anh Phu,19,M,Programer
24015,Nguyen Khanh Duy,23,M,Programer
24016,Tran Thi Hop,20,F,doctor
24017,Pham Van Thang,21,M,doctor
24018,Huynh Van Khoe,18,M,farmer
24019,Tran thi Be Phuong,17,F,Marketing
24020,To Kim Ngan,17,F,business
24021,Nguyen Minh Tam,19,M,officer
24022,Duong Trung Duong,20,M,web programming
24023,Nguyen Ngoc Khanh,21,M,developer
24024,Ha Minh Khang,20,M,technique
24025,Nguyen Thanh Tuong,19,M,Lawyer
24026,Nguyen Minh tien,22,M,IT worker
24027,Truong Nguyen Kieu Phuong,21,F,dentist
24028,Vo Vuong Hoa,22,M,Programer
24029,Dang Ngoc Ven,22,F,travel guide
24030,Bui Dinh Quoc Dong,20,M,electrician
24031,Nguyen Thi Thanh Hang,18,F,teacher
24032,Nguyen Tri,19,M,Programer
```

Picture 5.2.1 File User.txt



```
2401,2402
2401,2403
2401,2408
2402,2404
2402,2405
2403,2405
2403,2406
2403,2408
2404,2408
2404,24010
2405,24012
2405,24013
2406,24014
2406,24016
2407,24010
2407,24011
2407,24025
2408,2407
2409,24012
2409,24016
2409,24024
24010,24011
24011,24022
24012,24016
24012,24026
24013,24015
24013,24016
24013,24017
24014,24015
24015,24016
24015,24017
24016,24017
24017,24019
24017,24020
24018,24020
24019,24020
24019,24028
24019,24031
24020,24027
24020,24028
24021,24020
24022,24024
24022,24025
24023,2409
24024,24025
24024,24026
24025,24026
24026,24027
24027,24028
24027,24029
24028,24031
24029,24030
24030,24031
24031,24032
24032,24030
```

Picture 5.2.2 File UserEdge.txt

File User.java be below:

```
public class User{
    private int userID;
    private String userName;
    private int age;
    private String gender;
    private String job;
    public User(){
        this.userID =0;
        this.userName ="";
        this.age = 0;
        this.gender = "";
        this.job = "";
    }
    public User(int userID, String userName, int age, String gender, String
job) {
        this.userID = userID;
        this.userName = userName;
        this.age = age;
        this.gender = gender;
        this.job = job;
    }
    public int getUserID() {
        return userID;
    }
    public void setUserID(int userID) {
        this.userID = userID;
    }
    public String getUserName() {
        return userName;
    }
    public void setUserName(String userName) {
        this.userName = userName;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public String getGender() {
        return gender;
    }
    public void setGender(String gender) {
        this.gender = gender;
    }
    public String getJob() {
        return job;
    }
    public void setJob(String job) {
        this.job = job;
    }
    @Override
    public String toString() {
        return "User[" + getUserID() + "," + getUserName() + "," + getAge()
+ "," + getGender() + "," + getJob() + "]";
    }
}
```

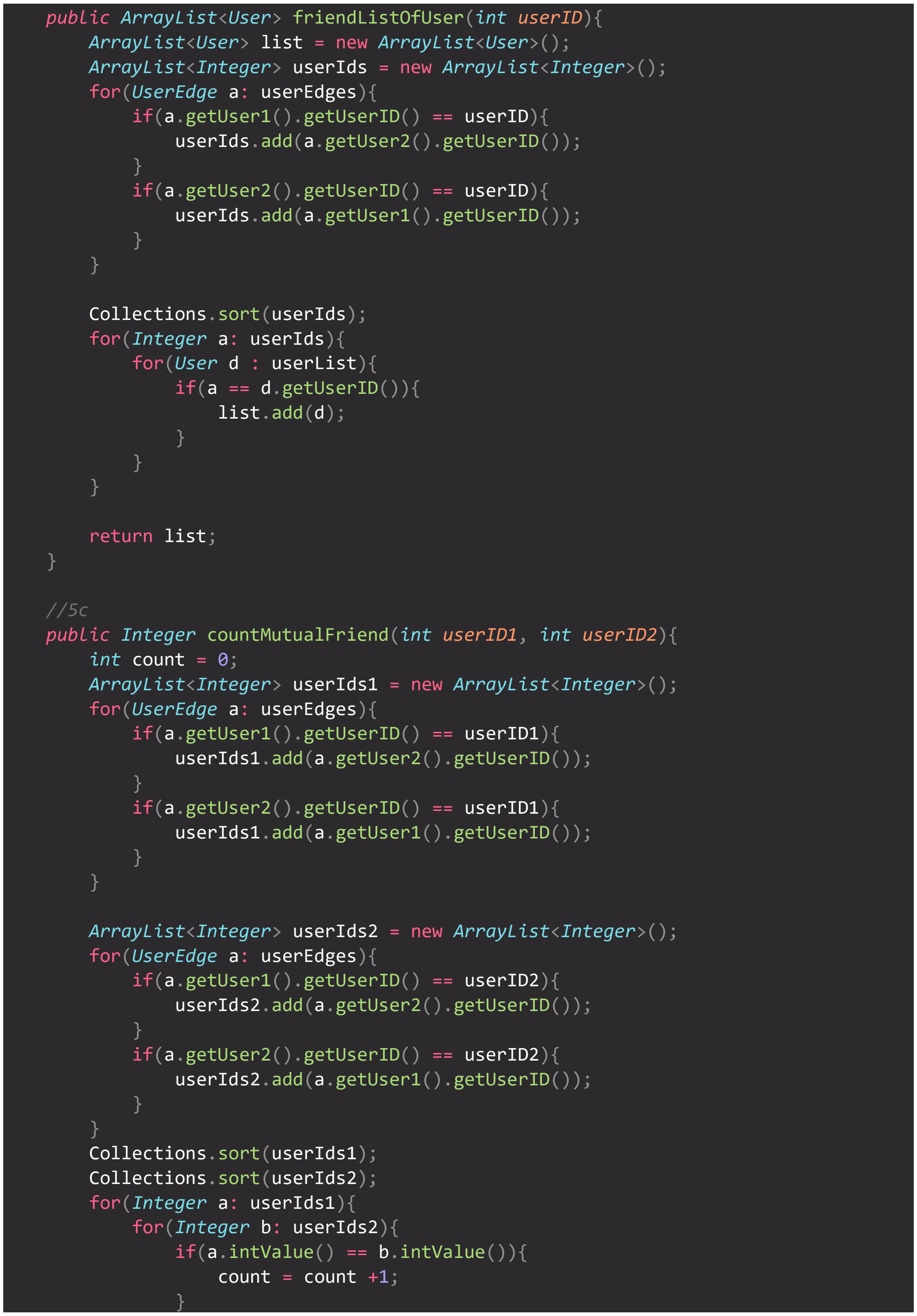
File UserEdge.java be below:

```
public class UserEdge {
    User user1;
    User user2;
    public UserEdge(User user1,User user2) {
        setUser1(user1);
        setUser2(user2);
    }
    public User getUser1() {
        return this.user1;
    }
    public void setUser1(User user1) {
        this.user1 = user1;
    }
    public User getUser2() {
        return this.user2;
    }
    public void setUser2(User user2) {
        this.user2 = user2;
    }
    @Override
    public String toString() {
        return  user1 +"," + user2;
    }
}
```

File UserManagement.java be below:

```
import java.util.*;
import java.io.*;
public class UserManagement {
    private ArrayList<User> userList;
    private ArrayList<UserEdge> userEdges;
    public UserManagement(String USerpath,String UserEdgepath){
        userList = new ArrayList<User>();
        userEdges = new ArrayList<UserEdge>();
        readUserFile(USerpath);
        CreateGraph(UserEdgepath);
    }
    public boolean readUserFile(String filepath){
        try{
```

```java
            File file = new File(filepath);
            Scanner sc = new Scanner(file);
            while(sc.hasNextLine()){
                String row = sc.nextLine();
                String[] attributes = row.split(",");
                User tmpUser = new User(Integer.parseInt(attributes[0]),
attributes[1], Integer.parseInt(attributes[2]), attributes[3],attributes[4]);
                userList.add(tmpUser);
            }
            sc.close();
        }catch(FileNotFoundException  e){
            System.out.println("file's not found!");
            return false;
        }
        return true;
    }
    public void addEgde(User user1, User user2){
        userEdges.add(new UserEdge(user1, user2));
    }
    public boolean CreateGraph(String filepath){
        try{
            File file= new File(filepath);
            Scanner scanner= new Scanner(file);
            while(scanner.hasNextLine()){
                User user1 = new User();
                User user2 = new User();
                String ch = scanner.nextLine();
                String[] attributes = ch.split(",");
                for(User u : userList){
                    if(u.getUserID() == Integer.parseInt(attributes[0])){
                        user1 = u;
                    }
                }
                for(User u : userList){
                    if(u.getUserID() == Integer.parseInt(attributes[1])){
                        user2 = u;
                    }
                }
                addEgde(user1, user2);
            }
            scanner.close();
        }catch(FileNotFoundException e){
            System.out.println("file's not found");
            return false;
        }
        return true;
    }
    public void printGraph(){
        for(UserEdge s : userEdges){
            System.out.println(s);
        }
    }
    //5b
```



```
public ArrayList<User> friendListOfUser(int userID){
    ArrayList<User> list = new ArrayList<User>();
    ArrayList<Integer> userIds = new ArrayList<Integer>();
    for(UserEdge a: userEdges){
        if(a.getUser1().getUserID() == userID){
            userIds.add(a.getUser2().getUserID());
        }
        if(a.getUser2().getUserID() == userID){
            userIds.add(a.getUser1().getUserID());
        }
    }

    Collections.sort(userIds);
    for(Integer a: userIds){
        for(User d : userList){
            if(a == d.getUserID()){
                list.add(d);
            }
        }
    }

    return list;
}

//5c
public Integer countMutualFriend(int userID1, int userID2){
    int count = 0;
    ArrayList<Integer> userIds1 = new ArrayList<Integer>();
    for(UserEdge a: userEdges){
        if(a.getUser1().getUserID() == userID1){
            userIds1.add(a.getUser2().getUserID());
        }
        if(a.getUser2().getUserID() == userID1){
            userIds1.add(a.getUser1().getUserID());
        }
    }

    ArrayList<Integer> userIds2 = new ArrayList<Integer>();
    for(UserEdge a: userEdges){
        if(a.getUser1().getUserID() == userID2){
            userIds2.add(a.getUser2().getUserID());
        }
        if(a.getUser2().getUserID() == userID2){
            userIds2.add(a.getUser1().getUserID());
        }
    }
    Collections.sort(userIds1);
    Collections.sort(userIds2);
    for(Integer a: userIds1){
        for(Integer b: userIds2){
            if(a.intValue() == b.intValue()){
                count = count +1;
            }
            }
        }
        return count;
    }
}
```

File MainUser.java is below:

```
import java.util.*;
public class MainUser {
    public static void main(String[] args) {
        UserManagement userMana = new UserManagement("User.txt","UserEdge.txt");
        userMana.printGraph();

        System.out.println();

        ArrayList<User> list = userMana.friendListOfUser(24016); // id of user 16
        System.out.println("the neighbors of user which has UserID as 24016:");
        for(User u : list) {
            System.out.println(u);
        }

        System.out.println();

        int count = userMana.countMutualFriend(24012,24013); //id of user 12 and
user 13
        System.out.println("Amount the mutual friend of two user: "+count);
    }
}
```