Rusyaidi Bin Imran Yeng 19001549 CoE

TDB1023: Algorithm and Data Structure

Certification of Originality

I hereby confirm and understand that :

☑ Plagiarism (copying / cheating) is not permitted, and if caught and found guilty, I will receive an F grade for this extended assignment, and will be subjected to the Academic Disciplinary Committee.

☑ All answers submitted for this extended assignment is my own original work.

Signature:

Rusyaidi Bin Imran Yeng

Student's Name:

1.a) There are generally two data structure methods that can be used in this scenario. The two of which are linked list and array. In my opinion, the most suitable data structure method that can be used for this program is linked list as linked list has a lot of functions that cannot be done using array.

Firstly, this is because linked list is a dynamic data structure. This means that it can grow and shrink at runtime by allocating and deallocating memory. Unlike array, the user will not need to declare an initial size to linked list. By declaring a size, it will also set a limited amount of data space that can be used. This will also waste the user's time as you need to change the array size every time you add a new data.

Besides that, linked list is easier to implement the two functions of insertion and deletion as the program requires be able to add and delete new and existing employee record. In linked list, the system only has to update the address of the next node. In array, the user must shift the elements after insertion or deletion of another element. Using an array will also consume a lot of time during the operation of insertion and deletion. On the other hand, the performance of these operations in Linked lists is fast.

There is also less memory wastage in linked list as it can increase and decrease at runtime. In an array, if the user declares 5 elements for the size of the array and only uses 2 elements there will be a wastage of 3 array elements.

1.b)

```
import java.util.Scanner;


public class Main {
        public void add(String Name, String ID, String bd, String designation, String paygrade, String phoneNum, String Address) {
                    //add method
 }
        public void search(String newName) {
                    //search method
                    //asks user for ID
                   // if ID = ID in linked list then only can access search method
                   // error message
       }
          public void delete(String name) {
                  //remove method
       }
```

```
    public void edit(String newName) {
        //edit method
    }
public void printall(){
        //print all data method
 }

public LinkedList() {
        //declares head and tail
}

public void insert(Employee emp_in) {
        //insert method
}

void delete(Employee emp_rem) {
        // delete method
      }
```

```
public void setName(String name) {
	//set method for name
}
public String getName() {
	//get method for name
}
public void setID(String ID) {
	//set method for ID
}
public String getID() {
	//get method for ID
}
public void setbd(String bd) {
	//set method for date of birth
}
public String getbd() {
	//get method for date of birth
}
```

```
public void setdesignation(String designation) {
        //set method for designation
 }
 public String getdesignation () {
        //get method for designation
 }
 public void setpaygrade(int paygrade) {
        //set method for paygrade
 }
 public String get paygrade () {
        //get method for paygrade
 }
 public void setphoneNum(String phoneNum) {
       //set method for contact number
 }
 public String getphoneNum() {
        //get method for contact number
 }
```

```
    public void setAddress(String Address) {
        //set method for Address
    }
    public String getAddress(){
        //get method for Address
    }
public static void main(String[] args) {
    while(true){
       //Print out options for user to choose
       choice){
          //case 1: add new employee
         // case 2: search employee
        // case 3: edit employee
       // case 4: delete employee
       // case 5: print all
       // case 6: quit
                }
    }
}
```

2.a) There are numerous data structure methods that can be used for this scenario, for example stack. In this case, queue linked list has been chosen as it seems the most suitable and fit for this task. In my opinion, queue linked list is the best because the doctor in the question prefers serving the patients in their arrival sequence and in a specific order which is female minor, male minor, adult woman and adult male.

By using queue, it allows the user to do a “first come first serve” scenario as queue will always implement First In, First Out rule in the program. FIFO will only remove the first node that has been created and will only add nodes from the back. This will make the insertion and deletion process easier as patients constantly walk in and walk out the hospital. Stack on the other hand can only remove the last node that has been created which follows the Last In First Out rule which is not suitable for this program.

Besides that similar to Linked List, queue has infinite length compared to fixed-size array. With that, the user does not have to go through the code just to change the size of the data.

Finally, queue is also fast and flexible as it can handle multiple data types .

Pairing queue and linked list will give both advantages mentioned in 1.a) and 2.a). Theerefore, making it a good data structure method.

2.b)

```
import java.util.* ;


class Main
{


static class Node {
   int data;


   int priority;


    Node next;


}


static Node node = new Node();
```

```
static Node newNode(int x, int y)
{
    Node temp = new Node();
    temp.data = x;
    temp.priority = y;
    temp.next = null;

    return temp;
}

static int peek(Node head)
{
    return (head).data;
}

static Node pop(Node head)
{
    if (head==null)
        System.out.println("Hall is empty.");
```

```
    Node temp = head;
    (head)  = (head).next;
    return head;
}

static Node push(Node head, int x, int y)
{
    Node start = (head);

    // Create new Node
    Node temp = newNode(x, y);


    if ((head).priority > y) {

        temp.next = head;
        (head) = temp;
    }
```

```
else {

        while (start.next != null &&
                start.next.priority < y) {
            start = start.next;
        }

        temp.next = start.next;
        start.next = temp;
    }
    return head;

}
static int isEmpty(Node head)
{
    return ((head) == null)?1:0;
}

public static void main(String args[])
{
    int i,ctr;
```

```
Node pq = newNode(0,0);


for(i=1; i<5; i++){
    int prioritySetter;


    Scanner in =new Scanner(System.in);


    System.out.println("Please enter '0' if you are a female under 18 ");
    System.out.println("Please enter '1' if you are a male under 18 ");
    System.out.println("Please enter '2' if you are a female above 18 ");
    System.out.println("Please enter '3' if you are a male above 18 \n");




    System.out.printf("Patient Number %d: ",i);
    prioritySetter=in.nextInt();
    System.out.printf("\n");
    if(prioritySetter>3){
        System.out.println("Invalid command.");
    }
```

```
        pq =push(pq, i, prioritySetter);
    }

    System.out.println("(Patient 0 is a dummy)");
    for(i=0; i<5; i++){

        System.out.printf("Based on age/gender priority, it's patient number %d turn.\n",
peek(pq),i);
        System.out.printf("Patient number %d is meeting the doctor right now, please be patient...
\n", peek(pq));
        pq=pop(pq);
        System.out.printf("Patient number %d is next, please be ready.\n\n",peek(pq));
        }

    }
    }
```