

Câu 1: Binary search

| Step | LB | UB | Considering index | Considering data | Conclusion |
| --- | --- | --- | --- | --- | --- |
| 1 | 0 | 9 | 5 | 25 | Update UB |
| 2 | 0 | 4 | 2 | 75 | Update UB |
| 3 | 0 | 1 | 1 | 90 | Found! |

**Câu 2: SORT**

| **Index**    **Action** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **100** | **52** | **3** | **22** | **31** | **67** | **42** | **75** | **23** | **0** |
| **Swap B[0] with B[9]** | **0** | 52 | 3 | 22 | 31 | 67 | 42 | 75 | 23 | **100** |
| **Swap B[1] with B[2]** | 0 | **3** | **52** | 22 | 31 | 67 | 42 | 75 | 23 | 100 |
| **Swap B[2] with B[3]** | 0 | 3 | 22 | 52 | 31 | 67 | 42 | 75 | 23 | 100 |
| **Swap B[3] with B[8]** | 0 | 3 | 22 | 23 | 31 | 67 | 42 | 75 | 52 | 100 |
| **Swap B[5] with B[6]** | 0 | 3 | 22 | 23 | 31 | 42 | 67 | 75 | 52 | 100 |
| **Swap B[6] with B[8]** | 0 | 3 | 22 | 23 | 31 | 42 | 52 | 75 | 67 | 100 |
| **Swap B[7] with B[8]** | 0 | 3 | 22 | 23 | 31 | 42 | 52 | 67 | 75 | 100 |

**Câu 3: STACK**

**public class StackWithDoubleEndList implements IStack {**

**private IDoubleEndList list;**

**public StackWithDoubleEndList() {**

**this.list = new DoubleEndLinkedList();**

**}**

**public void push(int data) {**

**list.insertFirst(data);**

**}**

**public int pop() {**

**if (list.isEmpty()) {**

**throw new IllegalStateException("Stack is empty");**

**}**

**return list.deleteFirst();**

**}**

**public int peek() {**

**if (list.isEmpty()) {**

**throw new IllegalStateException("Stack is empty");**

**}**

**return list.getFirst();**

**}**

**public boolean isEmpty() {**

**return list.isEmpty();**

**}**

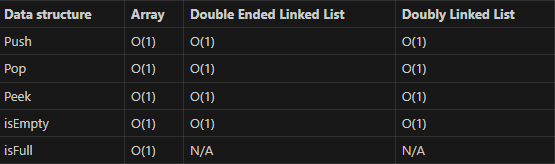
**public boolean isFull() {**

**return false;**

**}**

**}**

**Câu 4: Linked List**

****

Explanation:

- Array:

- Push: O(1) - Adding an element to the end of an array takes constant time.

- Pop: O(1) - Removing an element from the end of an array takes constant time.

- Peek: O(1) - Accessing the last element of an array takes constant time.

- isEmpty: O(1) - Checking if an array is empty takes constant time.

- isFull: O(1) - Checking if an array is full takes constant time.

- Double Ended Linked List:

- Push: O(1) - Adding an element to the end of a double ended linked list takes constant time.

- Pop: O(1) - Removing an element from the end of a double ended linked list takes constant time.

- Peek: O(1) - Accessing the last element of a double ended linked list takes constant time.

- isEmpty: O(1) - Checking if a double ended linked list is empty takes constant time.

- isFull: N/A - A double ended linked list does not have a fixed size, so there is no concept of being full.

- Doubly Linked List:

- Push: O(1) - Adding an element to the end of a doubly linked list takes constant time.

- Pop: O(1) - Removing an element from the end of a doubly linked list takes constant time.

- Peek: O(1) - Accessing the last element of a doubly linked list takes constant time.

- isEmpty: O(1) - Checking if a doubly linked list is empty takes constant time.

- isFull: N/A - A doubly linked list does not have a fixed size, so there is no concept of being full.

**Câu 5: Queue**

**import java.util.Stack;**

**public class Queue<T> {**

**private Stack<T> enqueueStack;**

**private Stack<T> dequeueStack;**

**public Queue() {**

**enqueueStack = new Stack<>();**

**dequeueStack = new Stack<>();**

**}**

**public void enqueue(T element) {**

**enqueueStack.push(element);**

**}**

**public T dequeue() {**

**if (dequeueStack.isEmpty()) {**

**while (!enqueueStack.isEmpty()) {**

**dequeueStack.push(enqueueStack.pop());**

**}**

**} return dequeueStack.pop();**

**}**

**}**

(Similar to Pair<T,K>)