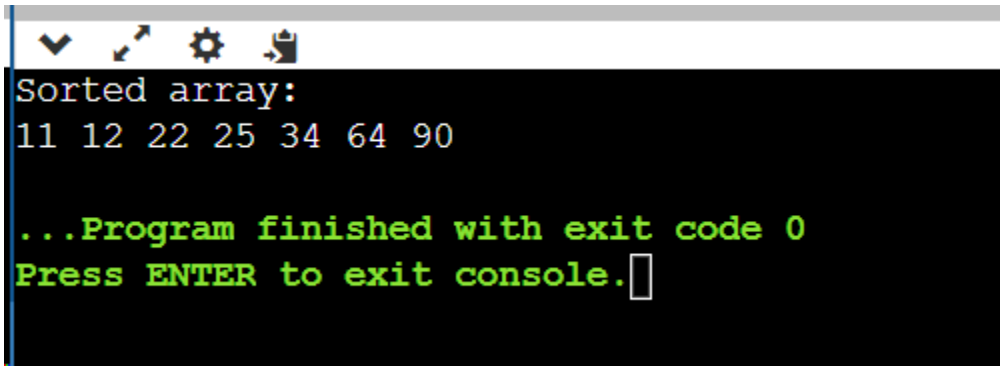
Logic Building Special Questions

Write implementation of below sorting algorithms

1. Bubble Sort

```
public class BubbleSort {
    public static void bubbleSort(int[] arr) {
        int n = arr.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    // Swap arr[j] and arr[j+1]
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }

    public static void main(String[] args) {
        int[] arr = {64, 34, 25, 12, 22, 11, 90};
        bubbleSort(arr);
        System.out.println("Sorted array:");
        for (int i : arr) {
            System.out.print(i + " ");
```

```
    }

  }

}
```

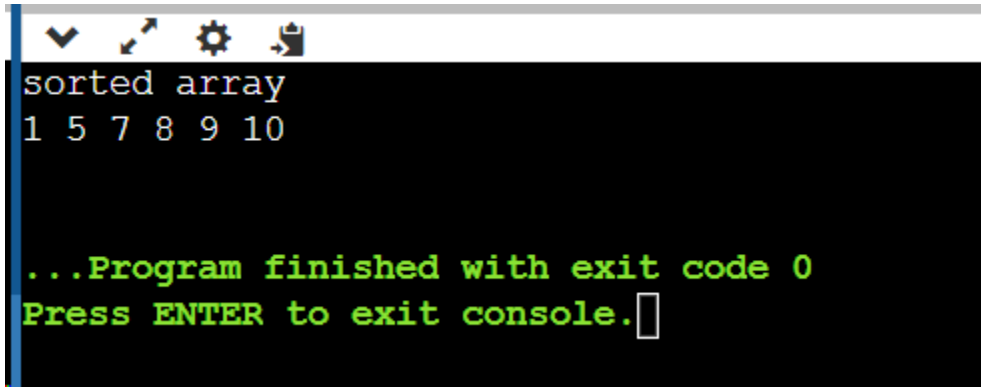


2. Quick Sort

```
// Java program for implementation of QuickSort
class Main
{

        int partition(int arr[], int low, int high)
        {
                int pivot = arr[high];
                int i = (low-1); // index of smaller element
                for (int j=low; j<high; j++)
                {
                        // If current element is smaller than or
                        // equal to pivot
                        if (arr[j] <= pivot)
                        {
                                i++;

                                // swap arr[i] and arr[j]
                                int temp = arr[i];
                                arr[i] = arr[j];
                                arr[j] = temp;
                        }
                }

                // swap arr[i+1] and arr[high] (or pivot)
                int temp = arr[i+1];
                arr[i+1] = arr[high];
```

```
        arr[high] = temp;

        return i+1;
}



void sort(int arr[], int low, int high)
{
        if (low < high)
        {
                /* pi is partitioning index, arr[pi] is
                now at right place */
                int pi = partition(arr, low, high);

                // Recursively sort elements before
                // partition and after partition
                sort(arr, low, pi-1);
                sort(arr, pi+1, high);
        }
}


static void printArray(int arr[])
{
        int n = arr.length;
        for (int i=0; i<n; ++i)
                System.out.print(arr[i]+" ");
        System.out.println();
}


public static void main(String args[])
{
        int arr[] = {10, 7, 8, 9, 1, 5};
        int n = arr.length;

        Main ob = new Main();
        ob.sort(arr, 0, n-1);

        System.out.println("sorted array");
        printArray(arr);
```

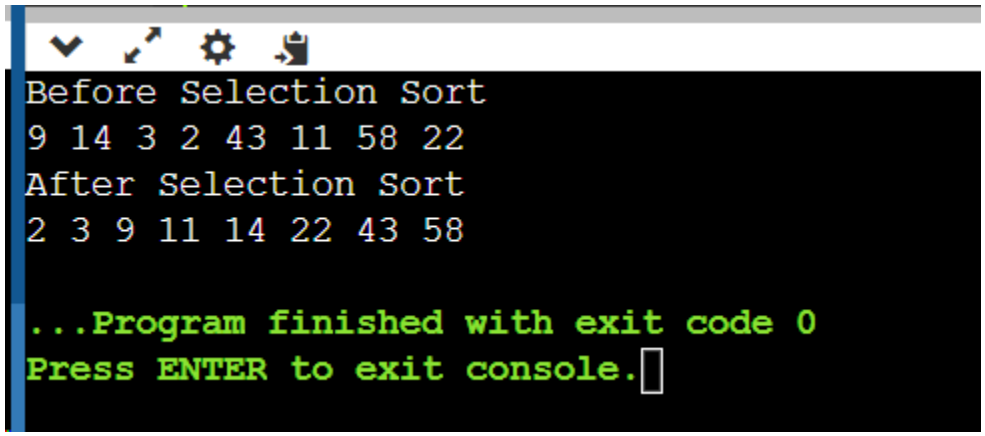
```
        }
}
```



3. Selection Sort

```
public class Main {
  public static void selectionSort(int[] arr){
    for (int i = 0; i < arr.length - 1; i++)
    {
      int index = i;
      for (int j = i + 1; j < arr.length; j++){
        if (arr[j] < arr[index]){
          index = j;//searching for lowest index
        }
      }
      int smallerNumber = arr[index];
      arr[index] = arr[i];
      arr[i] = smallerNumber;
    }
  }

  public static void main(String a[]){
    int[] arr1 = {9,14,3,2,43,11,58,22};
    System.out.println("Before Selection Sort");
    for(int i:arr1){
      System.out.print(i+" ");
    }
    System.out.println();

    selectionSort(arr1);//sorting array using selection sort

    System.out.println("After Selection Sort");
    for(int i:arr1){
      System.out.print(i+" ");
```

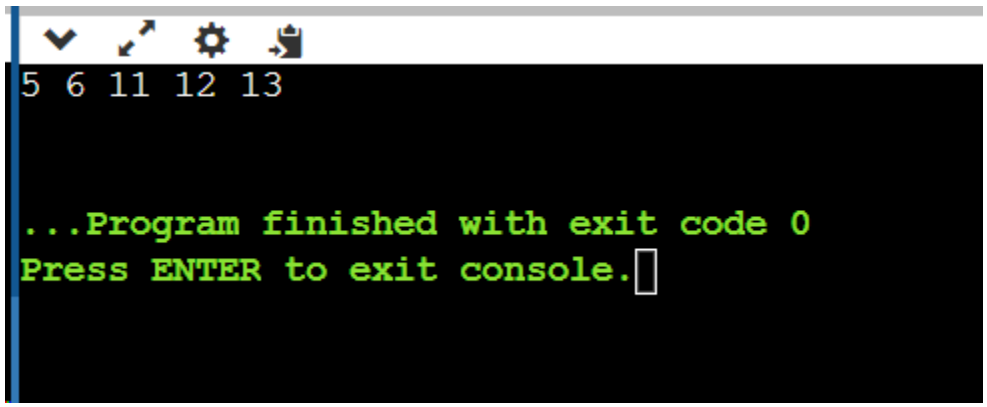
```
        }
    }
}
```



4. Insertion Sort

```
public class Main {
    void sort(int arr[])
    {
        int n = arr.length;
        for (int i = 1; i < n; ++i) {
            int key = arr[i];
            int j = i - 1;

            while (j >= 0 && arr[j] > key) {
                arr[j + 1] = arr[j];
                j = j - 1;
            }
            arr[j + 1] = key;
        }
    }

    static void printArray(int arr[])
    {
        int n = arr.length;
        for (int i = 0; i < n; ++i)
            System.out.print(arr[i] + " ");

        System.out.println();
    }

    public static void main(String args[])
```

```
    {
        int arr[] = { 12, 11, 13, 5, 6 };

        Main ob = new Main();
        ob.sort(arr);

        printArray(arr);
    }
};
```

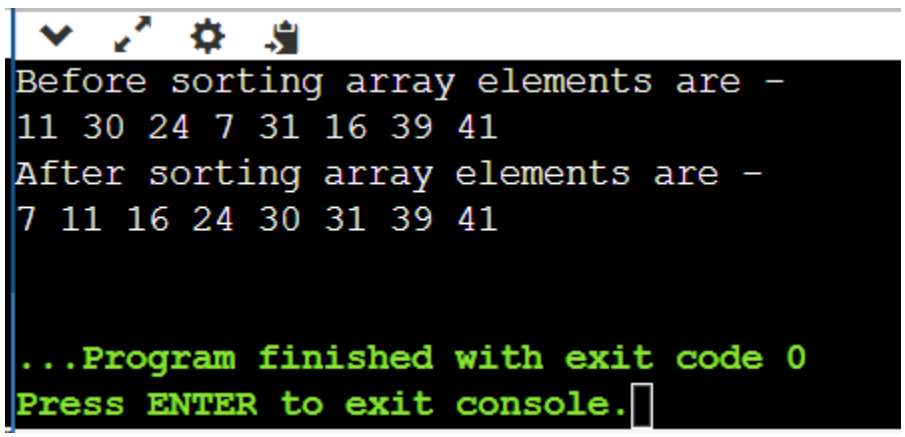


5. Merge Sort

```
class Main {

void merge(int a[], int beg, int mid, int end)
{
    int i, j, k;
    int n1 = mid - beg + 1;
    int n2 = end - mid;

        int LeftArray[] = new int[n1];
        int RightArray[] = new int[n2];

    for (i = 0; i < n1; i++)
    LeftArray[i] = a[beg + i];
    for (j = 0; j < n2; j++)
    RightArray[j] = a[mid + 1 + j];

    i = 0;
    j = 0;
    k = beg;

    while (i < n1 && j < n2)
```

```
    {
        if(LeftArray[i] <= RightArray[j])
        {
            a[k] = LeftArray[i];
            i++;
        }
        else
        {
            a[k] = RightArray[j];
            j++;
        }
        k++;
    }
    while (i<n1)
    {
        a[k] = LeftArray[i];
        i++;
        k++;
    }

    while (j<n2)
    {
        a[k] = RightArray[j];
        j++;
        k++;
    }
}

void mergeSort(int a[], int beg, int end)
{
    if (beg < end)
    {
        int mid = (beg + end) / 2;
        mergeSort(a, beg, mid);
        mergeSort(a, mid + 1, end);
        merge(a, beg, mid, end);
    }
}

void printArray(int a[], int n)
{
    int i;
```

```
    for (i = 0; i < n; i++)
        System.out.print(a[i] + " ");
}

public static void main(String args[])
{
    int a[] = { 11, 30, 24, 7, 31, 16, 39, 41 };
    int n = a.length;
    Main m1 = new Main();
    System.out.println("\nBefore sorting array elements are - ");
    m1.printArray(a, n);
    m1.mergeSort(a, 0, n - 1);
    System.out.println("\nAfter sorting array elements are - ");
    m1.printArray(a, n);
    System.out.println("");
}

 }
```

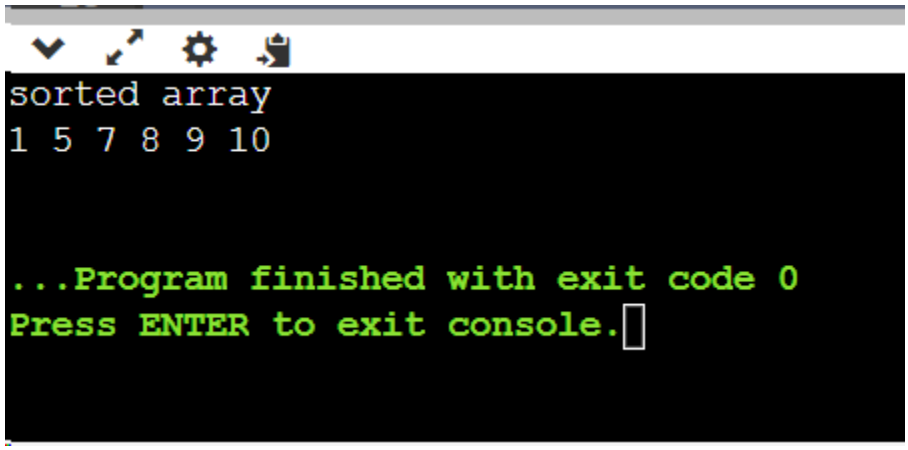


6. **Quick Sort**

```
// Java program for implementation of QuickSort
class Main
{

        int partition(int arr[], int low, int high)
        {
                int pivot = arr[high];
                int i = (low-1); // index of smaller element
                for (int j=low; j<high; j++)
                {
                        // If current element is smaller than or
                        // equal to pivot
```

```
                if (arr[j] <= pivot)
                {
                        i++;

                        // swap arr[i] and arr[j]
                        int temp = arr[i];
                        arr[i] = arr[j];
                        arr[j] = temp;
                }
        }

        // swap arr[i+1] and arr[high] (or pivot)
        int temp = arr[i+1];
        arr[i+1] = arr[high];
        arr[high] = temp;

        return i+1;
}



void sort(int arr[], int low, int high)
{
        if (low < high)
        {
                int pi = partition(arr, low, high);

                sort(arr, low, pi-1);
                sort(arr, pi+1, high);
        }
}
static void printArray(int arr[])
{
        int n = arr.length;
        for (int i=0; i<n; ++i)
                System.out.print(arr[i]+" ");
        System.out.println();
}

public static void main(String args[])
{
        int arr[] = {10, 7, 8, 9, 1, 5};
```

```
        int n = arr.length;

        Main ob = new Main();
        ob.sort(arr, 0, n-1);

        System.out.println("sorted array");
        printArray(arr);
    }
}
```



7. Sorting Strings using Bubble Sort

```
public class Main {
    public static void bubbleSort(String[] arr) {
        int n = arr.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                // Compare adjacent strings and swap if necessary
                if (arr[j].compareTo(arr[j + 1]) > 0) {
                    String temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }

    public static void main(String[] args) {
        String[] arr = {"banana", "apple", "orange", "grape", "pineapple"};

        // Print unsorted array
        System.out.println("Unsorted array:");
```

```
        for (String s : arr) {
            System.out.print(s + " ");
        }
        System.out.println();

        // Sort the array
        bubbleSort(arr);

        // Print sorted array
        System.out.println("Sorted array:");
        for (String s : arr) {
            System.out.print(s + " ");
        }
    }
}
```

```
Unsorted array:
banana apple orange grape pineapple
Sorted array:
apple banana grape orange pineapple

...Program finished with exit code 0
Press ENTER to exit console.
```

8. Bubble Sort for Linked List by Swapping nodes

```
class Node {
    int data;
    Node next;

    Node(int data) {
        this.data = data;
        this.next = null;
    }
}

public class Main {
    Node head;

    void append(int data) {
        if (head == null) {
```

```
        head = new Node(data);
        return;
    }
    Node current = head;
    while (current.next != null) {
        current = current.next;
    }
    current.next = new Node(data);
}

void bubbleSort() {
    if (head == null || head.next == null) {
        return;
    }
    boolean swapped;
    do {
        swapped = false;
        Node prev = null;
        Node current = head;
        Node nextNode = head.next;
        while (nextNode != null) {
            if (current.data > nextNode.data) {
                if (prev != null) {
                    prev.next = nextNode;
                } else {
                    head = nextNode;
                }
                current.next = nextNode.next;
                nextNode.next = current;
                swapped = true;
            }
            prev = current;
            current = nextNode;
            nextNode = nextNode.next;
        }
    } while (swapped);
}

void printList() {
    Node current = head;
    while (current != null) {
        System.out.print(current.data + " ");
```

```
            current = current.next;
        }
        System.out.println();
    }

    public static void main(String[] args) {
        Main list = new Main();
        list.append(64);
        list.append(34);
        list.append(25);
        list.append(12);
        list.append(22);
        list.append(11);
        System.out.println("Linked list before sorting:");
        list.printList();
        list.bubbleSort();
        System.out.println("Linked list after sorting:");
        list.printList();
    }
}
```

```
Linked list before sorting:
64 34 25 12 22 11
Linked list after sorting:
11 12 22 25 34 64


...Program finished with exit code 0
Press ENTER to exit console.
```

10.Bubble Sort On Doubly Linked List

```
class Node {

    int data;

    Node prev, next;



    Node(int data) {
```

```
        this.data = data;
        prev = next = null;
    }
}

public class Main {
    Node head;
    void append(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
            return;
        }
        Node current = head;
        while (current.next != null) {
            current = current.next;
        }
        current.next = newNode;
        newNode.prev = current;
    }
    void bubbleSort() {
        if (head == null || head.next == null) {
            return;
        }
        Node lastSorted = null;
```

```
    boolean swapped;
    do {
        swapped = false;
        Node current = head;
        while (current.next != lastSorted) {
            if (current.data > current.next.data) {
                swap(current, current.next);
                swapped = true;
            }
            current = current.next;
        }
        lastSorted = current;
    } while (swapped);
}
void swap(Node a, Node b) {
    int temp = a.data;
    a.data = b.data;
    b.data = temp;
}


void printList() {
    Node current = head;
    while (current != null) {
        System.out.print(current.data + " ");
        current = current.next;
```

```
        }
        System.out.println();
    }

    public static void main(String[] args) {
        Main list = new Main();
        list.append(64);
        list.append(34);
        list.append(25);
        list.append(12);
        list.append(22);
        list.append(11);
        System.out.println("Doubly linked list before sorting:");
        list.printList();
        list.bubbleSort();
        System.out.println("Doubly linked list after sorting:");
        list.printList();
    }
}
```

```
Doubly linked list before sorting:
64 34 25 12 22 11
Doubly linked list after sorting:
11 12 22 25 34 64


...Program finished with exit code 0
Press ENTER to exit console.
```