

LABSHEET 2

1. Print the first N natural numbers.

```
1 import java.util.Scanner;
2
3 class NaturalNumbers {
4     void printNatural(int n) {
5         if (n == 0) return;
6         printNatural(n - 1);
7         System.out.print(n + " ");
8     }
9 }
10
11 class Q1 {
12     Run main | Debug main | Run | Debug
13     public static void main(String[] args) {
14         Scanner sc = new Scanner(System.in);
15         NaturalNumbers nn = new NaturalNumbers();
16
17         System.out.println(x:"Enter the value of N: ");
18         int n = sc.nextInt();
19         nn.printNatural(n);
20     }
21 }
22
```

```
Enter the value of N:
10
1 2 3 4 5 6 7 8 9 10
```

2. Print the first N natural numbers in reverse order.

```
1 import java.util.Scanner;
2
3 class ReverseNaturalNumbers {
4     void printReverse(int n) {
5         if (n == 0) return;
6         System.out.print(n + " ");
7         printReverse(n - 1);
8     }
9 }
10
11 class Q2 {
12     Run main | Debug main | Run | Debug
13     public static void main(String[] args) {
14         Scanner sc = new Scanner(System.in);
15         ReverseNaturalNumbers rnn = new ReverseNaturalNumbers();
16
17         System.out.println(x:"Enter the value of N: ");
18         int n = sc.nextInt();
19         rnn.printReverse(n);
20     }
21 }
22
```

```
Enter the value of N:
10
10 9 8 7 6 5 4 3 2 1
```

3. Print the product of the first N natural numbers.

```
1  import java.util.Scanner;
2
3  class ProductNaturalNumbers {
4      int product(int n) {
5          if (n <= 1) return 1;
6          return n * product(n - 1);
7      }
8  }
9
10 class Q3 {
11     Run main | Debug main | Run | Debug
12     public static void main(String[] args) {
13         Scanner sc = new Scanner(System.in);
14         ProductNaturalNumbers pnn = new ProductNaturalNumbers();
15
16         System.out.println(x:"Enter the value of N: ");
17         int n = sc.nextInt();
18
19         System.out.println("Product of first " + n + " natural numbers: " + pnn.
20                             product(n));
21     }
22 }
```

Enter the value of N:

5

Product of first 5 natural numbers: 120

4. Print the Nth Fibonacci number.

```
1  import java.util.Scanner;
2
3  class Fibonacci {
4      int fibonacci(int n) {
5          if (n <= 1) return n;
6          return fibonacci(n - 1) + fibonacci(n - 2);
7      }
8  }
9
10 class Q4 {
11     Run main | Debug main | Run | Debug
12     public static void main(String[] args) {
13         Scanner sc = new Scanner(System.in);
14         Fibonacci fib = new Fibonacci();
15
16         System.out.println(x:"Enter the value of N: ");
17         int n = sc.nextInt();
18
19         System.out.println(n + "th Fibonacci number: " + fib.fibonacci(n));
20     }
21 }
```

Enter the value of N:

8

8th Fibonacci number: 21

5. Calculate x^y .

```
Q5.java > Language Support for Java(TM) by Red Hat > Q5
1  import java.util.Scanner;
2
3  class Power {
4      int power(int x, int y) {
5          if (y == 0) return 1;
6          return x * power(x, y - 1);
7      }
8  }
9
10 class Q5 {
11     Run main | Debug main | Run | Debug
12     public static void main(String[] args) {
13         Scanner sc = new Scanner(System.in);
14         Power p = new Power();
15
16         System.out.println(x:"Enter the base (x): ");
17         int x = sc.nextInt();
18
19         System.out.println(x:"Enter the exponent (y): ");
20         int y = sc.nextInt();
21
22         System.out.println(x + "^" + y + " = " + p.power(x, y));
23     }
24 }
```

Enter the base (x):
2
Enter the exponent (y):
3
2^3 = 8

6. Find the GCD(HCF) of two numbers.

```
Q6.java > Language Support for Java(TM) by Red Hat > Q6
1  import java.util.Scanner;
2
3  class GCD {
4      int gcd(int a, int b) {
5          if (b == 0) return a;
6          return gcd(b, a % b);
7      }
8  }
9
10 class Q6 {
11     Run main | Debug main | Run | Debug
12     public static void main(String[] args) {
13         Scanner sc = new Scanner(System.in);
14         GCD g = new GCD();
15
16         System.out.println(x:"Enter the first number: ");
17         int a = sc.nextInt();
18
19         System.out.println(x:"Enter the second number: ");
20         int b = sc.nextInt();
21
22         System.out.println("GCD of " + a + " and " + b + " is: " + g.gcd(a, b));
23     }
24 }
```

```
Enter the first number:
153
Enter the second number:
172
GCD of 153 and 172 is: 1
```

7. Print the elements of an array.

```
1  import java.util.Scanner;
2
3  class ArrayElements {
4      void printArray(int[] arr, int index) {
5          if (index == arr.length) return;
6          System.out.print(arr[index] + " ");
7          printArray(arr, index + 1);
8      }
9  }
10
11 class Q7 {
12     Run main | Debug main | Run | Debug
13     public static void main(String[] args) {
14         Scanner sc = new Scanner(System.in);
15         ArrayElements ae = new ArrayElements();
16
17         System.out.println(x:"Enter number of array elements: ");
18         int n = sc.nextInt();
19
20         int[] arr = new int[n];
21         for (int i = 0; i < n; i++) {
22             System.out.printf(format:"Enter element %d: ", i);
23             arr[i] = sc.nextInt();
24         }
25
26         System.out.println(x:"Array elements: ");
27         ae.printArray(arr, index:0);
28     }
29 }
```

```
Enter number of array elements:
5
Enter element 0: 1
Enter element 1: 2
Enter element 2: 3
Enter element 3: 4
Enter element 4: 5
Array elements:
1 2 3 4 5
```

8. Print the elements of an array in reverse order.

```
1  import java.util.Scanner;
2
3  class ReverseArray {
4      void printReverse(int[] arr, int index) {
5          if (index < 0) return;
6          System.out.print(arr[index] + " ");
7          printReverse(arr, index - 1);
8      }
9  }
10
11 class Q8 {
12     Run main | Debug main | Run | Debug
13     public static void main(String[] args) {
14         Scanner sc = new Scanner(System.in);
15         ReverseArray ra = new ReverseArray();
16
17         System.out.println(x:"Enter number of array elements: ");
18         int n = sc.nextInt();
19
20         int[] arr = new int[n];
21         for (int i = 0; i < n; i++) {
22             System.out.printf(format:"Enter element %d: ", i);
23             arr[i] = sc.nextInt();
24         }
25
26         System.out.println(x:"Array elements in reverse order: ");
27         ra.printReverse(arr, n - 1);
28     }
29 }
```

```
Enter number of array elements:
5
Enter element 0: 6
Enter element 1: 7
Enter element 2: 19
Enter element 3: 20
Enter element 4: 27
Array elements in reverse order:
27 20 19 7 6
```

9. Reverse a given number.

```
1  import java.util.Scanner;
2
3  class ReverseNumber {
4      int reverse(int num, int rev) {
5          if (num == 0) return rev;
6          rev = rev * 10 + (num % 10);
7          return reverse(num / 10, rev);
8      }
9  }
10
11 class Q9 {
12     public static void main(String[] args) {
13         Scanner sc = new Scanner(System.in);
14         ReverseNumber rn = new ReverseNumber();
15
16         System.out.println(x:"Enter a number to reverse: ");
17         int num = sc.nextInt();
18
19         System.out.println("Reversed number: " + rn.reverse(num, rev:0));
20     }
21 }
```

```
Enter a number to reverse:
12345
Reversed number: 54321
```


10. Check if an array is sorted or not.

```
1  import java.util.Scanner;
2
3  class SortedArray {
4      boolean isSorted(int[] arr, int index) {
5          if (index == arr.length - 1) return true;
6          if (arr[index] > arr[index + 1]) return false;
7          return isSorted(arr, index + 1);
8      }
9  }
10
11 class Q10 {
12     Run main | Debug main | Run | Debug
13     public static void main(String[] args) {
14         Scanner sc = new Scanner(System.in);
15         SortedArray sa = new SortedArray();
16
17         System.out.println(x:"Enter number of array elements: ");
18         int n = sc.nextInt();
19
20         int[] arr = new int[n];
21         for (int i = 0; i < n; i++) {
22             System.out.printf(format:"Enter element %d: ", i);
23             arr[i] = sc.nextInt();
24         }
25
26         if (sa.isSorted(arr, index:0))
27             System.out.println(x:"Array is sorted.");
28         else
29             System.out.println(x:"Array is not sorted.");
30     }
31 }
```

```
Enter number of array elements:
5
Enter element 0: 23
Enter element 1: 43
Enter element 2: 12
Enter element 3: -34
Enter element 4: 56
Array is not sorted.
```

11. Write a recursive algorithm to find the median of an array in $O(n)$ time.

```
import java.util.Scanner;
class MedianFinder {
    int partition(int[] arr, int low, int high) {
        int pivot = arr[high];
        int i = low;

        for (int j = low; j < high; j++) {
            if (arr[j] <= pivot) {
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
                i++;
            }
        }
        int temp = arr[i];
        arr[i] = arr[high];
        arr[high] = temp;
        return i;
    }

    int quickSelect(int[] arr, int low, int high, int k) {
        if (low <= high) {
            int pivotIndex = partition(arr, low, high);

            if (pivotIndex == k) {
                return arr[pivotIndex];
            } else if (pivotIndex > k) {
                return quickSelect(arr, low, pivotIndex - 1, k);
            } else {
                return quickSelect(arr, pivotIndex + 1, high, k);
            }
        }
        return Integer.MAX_VALUE;
    }

    int findMedian(int[] arr) {
        int n = arr.length;
        if (n % 2 == 1) {
            return quickSelect(arr, low:0, n - 1, n / 2);
        } else {
            int mid1 = quickSelect(arr, low:0, n - 1, n / 2 - 1);
            int mid2 = quickSelect(arr, low:0, n - 1, n / 2);
            return (mid1 + mid2) / 2;
        }
    }
}

class Q11 {
    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        MedianFinder mf = new MedianFinder();

        System.out.println("Enter number of array elements: ");
        int n = sc.nextInt();

        int[] arr = new int[n];
        for (int i = 0; i < n; i++) {
            System.out.printf("Enter element %d: ", i);
            arr[i] = sc.nextInt();
        }

        int median = mf.findMedian(arr);
        System.out.println("Median of the array: " + median);
    }
}
```

```
Enter number of array elements:
6
Enter element 0: 12
Enter element 1: 21
Enter element 2: 23
Enter element 3: 32
Enter element 4: 54
Enter element 5: 72
Median of the array: 27
```


12. Write a recursive algorithm to find the kth largest element in an array.

```
1  import java.util.Scanner;
2
3  class KthLargestElement {
4      int partition(int[] arr, int low, int high) {
5          int pivot = arr[high];
6          int i = low;
7
8          for (int j = low; j < high; j++) {
9              if (arr[j] >= pivot) {
10                 int temp = arr[i];
11                 arr[i] = arr[j];
12                 arr[j] = temp;
13                 i++;
14             }
15         }
16         int temp = arr[i];
17         arr[i] = arr[high];
18         arr[high] = temp;
19         return i;
20     }
21
22     int quickSelect(int[] arr, int low, int high, int k) {
23         if (low <= high) {
24             int pivotIndex = partition(arr, low, high);
25
26             if (pivotIndex == k) {
27                 return arr[pivotIndex];
28             } else if (pivotIndex > k) {
29                 return quickSelect(arr, low, pivotIndex - 1, k);
30             } else {
31                 return quickSelect(arr, pivotIndex + 1, high, k);
32             }
33         }
34         return Integer.MAX_VALUE;
35     }
36
37     int findKthLargest(int[] arr, int k) {
38         return quickSelect(arr, 0, arr.length - 1, k - 1);
39     }
40 }
41
42 class Q12 {
43     Run | Debug | Run main | Debug main
44     public static void main(String[] args) {
45         Scanner sc = new Scanner(System.in);
46         KthLargestElement kle = new KthLargestElement();
47
48         System.out.println("Enter number of array elements: ");
49         int n = sc.nextInt();
50
51         int[] arr = new int[n];
52         for (int i = 0; i < n; i++) {
53             System.out.printf("Enter element %d: ", i);
54             arr[i] = sc.nextInt();
55         }
56
57         System.out.println("Enter the value of k: ");
58         int k = sc.nextInt();
59
60         int kthLargest = kle.findKthLargest(arr, k);
61         System.out.println(k + "th largest element in the array: " + kthLargest);
62     }
63 }
64
```

```
Enter number of array elements:
5
Enter element 0: 12
Enter element 1: 34
Enter element 2: 56
Enter element 3: 67
Enter element 4: 78
Enter the value of k:
3
3th largest element in the array: 56
```

13 . Bubble Sort

```
1  import java.util.Scanner;
2
3  class BubbleSort {
4      void sort(int[] arr) {
5          int n = arr.length;
6          for (int i = 0; i < n - 1; i++) {
7              boolean swapped = false;
8              for (int j = 0; j < n - i - 1; j++) {
9                  if (arr[j] > arr[j + 1]) {
10                     int temp = arr[j];
11                     arr[j] = arr[j + 1];
12                     arr[j + 1] = temp;
13                     swapped = true;
14                 }
15             }
16             if (!swapped) break;
17         }
18     }
19 }
20
21 class Q13 {
22     Run | Debug | Run main | Debug main
23     public static void main(String[] args) {
24         Scanner sc = new Scanner(System.in);
25         BubbleSort bs = new BubbleSort();
26
27         System.out.print(s:"Enter the number of elements: ");
28         int n = sc.nextInt();
29         int[] arr = new int[n];
30
31         System.out.println(x:"Enter the elements:");
32         for (int i = 0; i < n; i++) {
33             arr[i] = sc.nextInt();
34         }
35
36         bs.sort(arr);
37
38         System.out.println(x:"Sorted array:");
39         for (int num : arr) {
40             System.out.print(num + " ");
41         }
42     }
43 }
```

```
Enter the number of elements: 5
Enter the elements:
7
9
-3
4
0
Sorted array:
-3 0 4 7 9
```

a. Time Complexity:

- **Worst Case:** $O(n^2)$ when the array is in reverse order.
- **Best Case:** $O(n^2)$ in the simple implementation, but can be improved to $O(n)$ with an optimization.

b. Optimization to Best Case $O(n)$:

- By adding a boolean flag to check if any swaps were made in a pass, we can stop early if the array is already sorted.
- **Worst Case Complexity:** $O(n^2)$.

c. Examples:

- **Best Case:** Already sorted array, e.g., [1, 2, 3, 4, 5]
- **Worst Case:** Reverse sorted array, e.g., [5, 4, 3, 2, 1]

14 . Selection Sort

```
1  import java.util.Scanner;
2
3  class SelectionSort {
4      void sort(int[] arr) {
5          int n = arr.length;
6          for (int i = 0; i < n - 1; i++) {
7              int minIndex = i;
8              for (int j = i + 1; j < n; j++) {
9                  if (arr[j] < arr[minIndex]) {
10                     minIndex = j;
11                 }
12             }
13             int temp = arr[minIndex];
14             arr[minIndex] = arr[i];
15             arr[i] = temp;
16         }
17     }
18 }
19 class Q14 {
20     Run | Debug | Run main | Debug main
21     public static void main(String[] args) {
22         Scanner sc = new Scanner(System.in);
23         SelectionSort ss = new SelectionSort();
24
25         System.out.print(s:"Enter the number of elements: ");
26         int n = sc.nextInt();
27         int[] arr = new int[n];
28
29         System.out.println(x:"Enter the elements:");
30         for (int i = 0; i < n; i++) {
31             arr[i] = sc.nextInt();
32         }
33
34         ss.sort(arr);
35
36         System.out.println(x:"Sorted array:");
37         for (int num : arr) {
38             System.out.print(num + " ");
39         }
40     }
```

```
Enter the number of elements: 6
Enter the elements:
3 8 2 -11 0 2
Sorted array:
-11 0 2 2 3 8
```

a. Time Complexity:

- **Worst Case:** $O(n^2)$
- **Best Case:** $O(n^2)$ because it always scans the unsorted part of the array.

b. Optimization to Best Case $\Omega(n)\Omega(n)\Omega(n)$:

- Selection Sort cannot be improved to $O(n)$ because it always checks the remaining unsorted elements.
- **Worst Case Complexity:** $O(n^2)$.

c. Examples:

- **Best Case:** Already sorted array, e.g., [1, 2, 3, 4, 5]
- **Worst Case:** Reverse sorted array, e.g., [5, 4, 3, 2, 1]

15. Insertion Sort

a. Time Complexity:

- **Worst Case:** $O(n^2)$ when the array is in reverse order.
- **Best Case:** $O(n)$ when the array is already sorted.

b. Optimization to Best Case $\Omega(n)$:

- Insertion Sort is naturally $O(n)$ in the best case when no shifting is needed (already sorted array).
- **Worst Case Complexity:** $O(n^2)$.

c. Examples:

- **Best Case:** Already sorted array, e.g., [1, 2, 3, 4, 5]
- **Worst Case:** Reverse sorted array, e.g., [5, 4, 3, 2, 1]

```

import java.util.Scanner;
class InsertionSort {
    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--;
            }
            arr[j + 1] = key;
        }
    }
}

class Q15 {
    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        InsertionSort is = new InsertionSort();

        System.out.print(s:"Enter the number of elements: ");
        int n = sc.nextInt();
        int[] arr = new int[n];

        System.out.println(x:"Enter the elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        is.sort(arr);

        System.out.println(x:"Sorted array:");
        for (int num : arr) {
            System.out.print(num + " ");
        }
    }
}

```

```

Enter the number of elements: 5
Enter the elements:
55 43 23 65 72
Sorted array:
23 43 55 65 72

```

16 .

Bit Manipulation

Difficulty: Easy Accuracy: 49.84% Submissions: 58K+ Points: 2 Average Time: 15m

Given a 32 bit unsigned integer **num** and an integer **i**. Perform following operations on the number -

1. **Get** ith bit
2. **Set** ith bit
3. **Clear** ith bit

Note : For better understanding, we are starting bits from 1 instead 0. (1-based). You have

Output Window

Compilation Results Custom Input Y.O.G.I. (AI Bot)

Problem Solved Successfully

[Suggest Feedback](#)

Test Cases Passed	Attempts : Correct / Total
20 / 20	1 / 1
	Accuracy : 100%

17 .

Count total set bits

Difficulty: Medium Accuracy: 35.77% Submissions: 216K+ Points: 4

You are given a number **n**. Find the total count of set bits for all numbers from 1 to n (both inclusive).

Examples :

Input: n = 4
Output: 5

Output Window

Compilation Results Custom Input Y.O.G.I. (AI Bot)

Problem Solved Successfully

[Suggest Feedback](#)

Test Cases Passed	Attempts : Correct / Total
1115 / 1115	1 / 1

18.

Odd or Even

Difficulty: Basic Accuracy: 60.6% Submissions: 130K+ Points: 1 Average Time: 5m

Given a positive integer **n**, determine whether it is odd or even. Return **true** if the number is even and **false** if the number is odd.

Examples:

Input: n = 15

Output: false

Explanation: The number is not divisible by 2

Output Window

Compilation Results

Custom Input

Y.O.G.I. (AI Bot)

Problem Solved Successfully

Suggest

Test Cases Passed

Attempts : Correct / Total

200 / 200

1 / 1