**Q1: Show, using the definition of the big-Oh, that the function f(n) = 0.5n + 255 is O(n)**

**Big-O(n)**

**n is valid for all value so,**

g(n) = 225/0.5 = 510

g(n) = n. For all n > 510

g(n) > f(n) **Therefore, f(n) = O(n).**

**Q2: Using the big-Oh notation, the worst-case running time of the following algorithm:**

Big-O:

The worst case of the algorithm **n3** because the three nested loop are working.

**n (**belong to outer loop**)** **\* n (**belong to inner loop**) \* n (**belong to mood condition**)**

**O (n3).**

**Q3: Describe the worst case running time of the following (a, b and c) codes in “big-Oh” notation in terms of the variable n.**

1. The worst case is O(n^4) => (n\*n\*n\*n) => n4 because of the nested loop.
2. The worst case is O(n) => (B and the function f3 are returning the values so the time complexity of returning the function is **O(n+n)** which is equal to **O(2n)**, so that removing constant so the **O(n)** )

**Q4 Suppose you have a queue Q, which contains two elements in the following order: 5 4.**

**A)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **5** | | **4** | | | |
| **5** | | **4** | | **3** | |
| **5** | **4** | | **3** | | **8** |
| **4** | **3** | | **8** | |  |
| **3** | **8** | |  | |  |
| **8** |  | |  | |  |
| **8** | **9** | |  | |  |

**Q.insert(3);**

**Q.insert(8);**

**Q.remove();**

**Q.remove();**

**Q.remove();**

**Q.insert(9);**

**B)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **5** | **4** | **3** |  |  |
| **5** | **4** | **3** | **8** |  |
| **4** | **3** | **8** |  |  |
| **3** | **8** |  |  |  |
| **8** |  |  |  |  |
|  |  |  |  |  |
| **1** |  |  |  |  |
| **1** | **2** |  |  |  |
| **1** | **2** | **3** |  |  |
| **1** | **2** | **3** | **4** |  |
| **1** | **2** | **3** | **4** | **5** |

**C) Code file is attached**

**Q5: What will be output of the following programs?**

The outer will run **< 3** and the inner loop will run **< I,**

**Stack output will be**

|  |
| --- |
| **B** |
| **A** |
| **C** |
| **A** |
| **B** |
| **A** |

**Queue Output will be**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **a** | **b** | **a** | **c** | **a** | **b** |

**Q6** **Design and analyze an algorithm that for a given positive integer n counts the number of**

**different ways to write n as a sum of 1, 2, 3, and 4. For examples, if n = 4, the output should be 8, as**

**Code:**

class GFG {  
 public static int count(int key){  
 int count = 0;  
 for(int i = 1; i <= 4; i++){  
 if(i == key)  
 count++;  
   
 for(int j = 1; j <= 4;j++){  
 if(i+j == key)  
 count++;  
   
 for(int k = 1; k <= 4; k++){  
 if(i + j + k == key)  
 count++;  
   
 for(int m = 1; m <= 4; m++){  
 if(i + j + k + m == key)  
 count++;  
   
 }  
 }  
 }  
 }  
 return count;  
 }  
 public static void main(String[] args) {  
 System.*out*.print(*count*(4));  
 }  
}

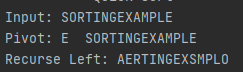
**output:**

**8**

**Q7: Write the code to satisfy the requirement, output should be as depicted in figure (attach**

**screenshots of the program output in word file and submit code as separate file)**

**Output:**

****

****

**Code:**

public class Quicksort {  
 static boolean *print* = true;  
 static int *pivot*;  
 public static void main(String[] args) {  
 String temp = "SORTINGEXAMPLE";  
  
 int[] arr = *ConvertIntemp*(temp);  
 int low = 0;  
 int high = arr.length-1;  
  
 System.*out*.println("---------- Quick Sort ----------");  
 System.*out*.println("Input: "+ temp);  
  
 *sort*(arr,low,high);  
  
  
 System.*out*.println("Sorted: "+*ConvertInString*(arr));  
 }  
  
 static int[] ConvertIntemp(String temp){  
 int[] arr = new int[temp.length()];  
  
 for (int i = 0; i < temp.length(); i++) {  
 arr[i] = Integer.*parseInt*(String.*valueOf*((int) temp.charAt(i)));  
 }  
  
 return arr;  
 }  
 static String ConvertInString(int[] temp){  
 String arr = "";  
  
 for (int i = 0; i < temp.length; i++) {  
 arr += String.*valueOf*((char) temp[i]);  
 }  
  
 return arr;  
 }  
  
  
  
 static void sort(int arr[], int low, int high){  
 if (low < high){  
 int pi = *partition*(arr, low, high);  
 if(true){  
 if(arr.length/2 <= low){  
 System.*out*.print("Recurse right: ");  
 }else {  
 System.*out*.print("Recurse Left: ");  
 }  
 System.*out*.print(*ConvertInString*(arr)+"\n");  
  
 }  
  
 *sort*(arr, low, pi-1);  
 *sort*(arr, pi+1, high);  
 }  
 }  
  
 static int partition(int arr[], int low, int high){  
 *pivot* = arr[high];  
 int i = (low-1); // index of smaller element  
  
 if(*print*){  
 *print* = false;  
 System.*out*.println("Pivot: "+((char) *pivot*)+" "+*ConvertInString*(arr));  
 }  
  
 for (int j=low; j<high; j++)  
 if (arr[j] < *pivot*)  
 i += *swap*(arr,i+1,j);  
  
  
 return (i+*swap*(arr,i+1,high));  
 }  
  
 public static int swap(int arr[], int i, int j) {  
 int temp = arr[i];  
 arr[i] = arr[j];  
 arr[j] = temp;  
  
 return 1;  
 }  
}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Q8. Evaluate and write the result for the following given expression, take values for the operands (show detailed work/dry run)**

**i) abc\*+de\*f+g\*+**

| **Input** | **Postfix** | **Infix** |
| --- | --- | --- |
| abc\*+de\*f+g\*+ | bc\*+de\*f+g\*+ | A |
| abc\*+de\*f+g\*+ | c\*+de\*f+g\*+ | Ab |
| abc\*+de\*f+g\*+ | \*+de\*f+g\*+ | Abc |
| abc\*+de\*f+g\*+ | +de\*f+g\*+ | a(b\*c) |
| abc\*+de\*f+g\*+ | de\*f+g\*+ | (a+(b\*c)) |
| abc\*+de\*f+g\*+ | e\*f+g\*+ | (a+(b\*c))d |
| abc\*+de\*f+g\*+ | \*f+g\*+ | (a+(b\*c))de |
| abc\*+de\*f+g\*+ | f+g\*+ | (a+(b\*c))(d\*e) |
| abc\*+de\*f+g\*+ | +g\*+ | (a+(b\*c))(d\*e)f |
| abc\*+de\*f+g\*+ | g\*+ | (a+(b\*c))((d\*e)+f) |
| abc\*+de\*f+g\*+ | \*+ | (a+(b\*c))((d\*e)+f)g |
| abc\*+de\*f+g\*+ | + | (a+(b\*c))(((d\*e)+f)\*g) |
| abc\*+de\*f+g\*+ |  | ((a+(b\*c))+(((d\*e)+f)\*g)) |
|  |  |  |

**((a+(b\*c))+(((d\*e)+f)\*g)) [**Now put the value a=1,b=2,c=3,d=4,e=5,f=6,g=7

(1+(2\*3))+(((4\*5)+6)\*7))

(1+6)+(20+6\*7)

7+182

**189**

**-+a\*/bc^d**

| **Input String** | **Prefix Expression** | **Stack (Infix)** |
| --- | --- | --- |
| -+a\*/bc\*def | -+a\*/bc\*de | f |
| -+a\*/bc\*def | -+a\*/bc\*d | fe |
| -+a\*/bc\*def | -+a\*/bc\* | fed |
| -+a\*/bc\*def | -+a\*/bc | f(d\*e) |
| -+a\*/bc\*def | -+a\*/b | f(d\*e)c |
| -+a\*/bc\*def | -+a\*/ | f(d\*e)cb |
| -+a\*/bc\*def | -+a\* | f(d\*e)(b/c) |
| -+a\*/bc\*def | -+a | f((b/c)\*(d\*e)) |
| -+a\*/bc\*def | -+ | f((b/c)8(d^e))a |
| -+a\*/bc\*def | - | f(a+((b/c)8(d^e))) |
| -+a\*/bc\*def |  | ((a+((b/c)\*(d^e)))-f) |

((a+((b/c)\*(d^e)))-f) **[**Now put the value a=1,b=2,c=3,d=4,e=5,f=6]

((1+((2/3)\*(4^5)))-6)

(1+((0.67)\*(1024)))-6

(1+682.6)-6

683.6-6

**Output: 677.6**

1. **( ( ( A + B ) \* C ) - ( ( D + E ) / F ) )**

Postfix: ab + c \* de + f /-

Prefix: -\*+abc/+def

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

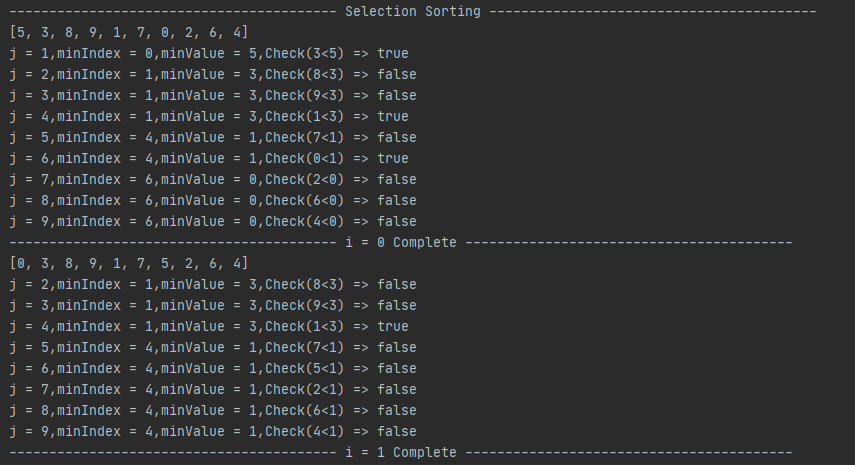
**Q9: Here is an array of ten integers:**

**5 3 8 9 1 7 0 2 6 4**

a) Write the output of this array after the FIRST iteration of the selection sort.

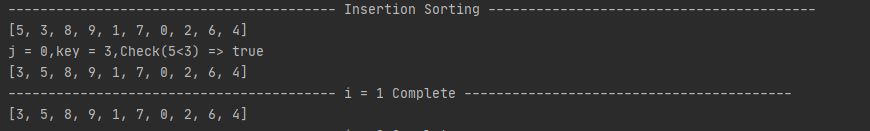
import java.util.Arrays;  
public class Main {  
 public static void main(String[] args) {  
 int array[] = {};  
  
 Sorting ob = new Sorting();  
 ob.SelectionSorting(array);  
 }  
}  
class Sorting{  
  
 void SelectionSorting(int arr[]){  
 int length = arr.length,  
 swap,  
 countloop = 0;  
 System.*out*.println("----------------------------------------- Selection Sorting ----------------------------------------- ");  
 System.*out*.println(Arrays.*toString*(arr));  
  
 for (int i = 0; i < (length-1); i++) {  
 int minIndex = i;  
 for (int j = i+1; j < length; j++) {  
 System.*out*.println("j = "+j+",minIndex = "+minIndex+",minValue = "+arr[minIndex]+",Check("+arr[j]+"<"+arr[minIndex]+") => "+(arr[j] < arr[minIndex]));  
 if (arr[j] < arr[minIndex]) {  
 minIndex = j;  
 }  
 countloop++;  
 }  
 swap = arr[minIndex];  
 arr[minIndex] = arr[i];  
 arr[i] = swap;  
 System.*out*.println("----------------------------------------- i = "+i+" Complete -----------------------------------------");  
 System.*out*.println(Arrays.*toString*(arr));  
  
 }  
  
 System.*out*.println("Number of time value inner loop works: "+countloop);  
 }

***Output:***



b) Then, write the output of this array after FIRST iteration of the loop in an insertion sort.

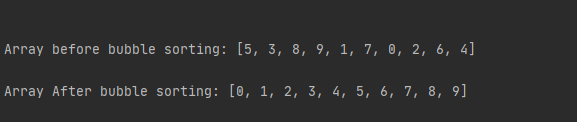
class Sorting{  
  
 void InsertionSorting(int arr[]) {  
 int length = arr.length,  
 countloop = 0;  
 System.*out*.println("----------------------------------------- Insertion Sorting ----------------------------------------- ");  
 System.*out*.println(Arrays.*toString*(arr));  
  
 for (int i = 1; i < (length); ++i) {  
 int key = arr[i];  
 int j = i - 1;  
 while (j >= 0 && arr[j] > key) {  
 System.*out*.println("j = "+j+",key = "+key+",Check("+arr[j]+"<"+key+") => "+(j >= 0 && arr[j] > key));  
 arr[j + 1] = arr[j];  
 j = j - 1;  
 countloop++;  
 }  
 arr[j + 1] = key;  
 System.*out*.println(Arrays.*toString*(arr));  
 System.*out*.println("----------------------------------------- i = " + i + " Complete -----------------------------------------");  
  
 }  
 System.*out*.println("Number of time value inner loop works: "+countloop);  
 }  
}

**output: **

c) Finally, sort the array using bubble sort.

package com.company;  
  
  
import java.util.Arrays;  
import java.util.Scanner;  
  
public class BubbleSorting1 {  
 public static void main(String[] args) {  
 System.*out*.println("1) Implement Bubble Sort.\n");  
  
// int length;  
// Scanner scan = new Scanner(System.in);  
  
// System.out.print("Enter the Length of the array: ");  
//  
 int arr[] = new int[]{5,3,8,9,1,7,0,2,6,4};  
 int length = arr.length;  
// for (int i = 0; i < length; i++) {  
// System.out.print("Enter the Value of index: "+i+" : ");  
// arr[i] = scan.nextInt();  
// }  
  
 System.*out*.println("\nArray before bubble sorting: "+ Arrays.*toString*(arr));  
  
 int swap;  
 for (int i = 0; i < length-1; i++) {  
 for (int j = 0; j < (length-i)-1; j++) {  
 if (arr[j] > arr[j+1]) {  
 swap = arr[j];  
 arr[j] = arr[j+1];  
 arr[j+1] = swap;  
 }  
 }  
 }  
  
 System.*out*.println("\nArray After bubble sorting: "+ Arrays.*toString*(arr));  
  
 }  
}

**Output:**

****

**Q10:** **Do as directed (Choose sorting algorithm wisely)**

**A)**

0 1 2 3 4 5 6 7 8 42 24 6 17 30 13 90 47 2 {I =1}

0 1 2 3 4 5 6 7 8 42 24 6 17 30 13 90 47 2 {I =2}

0 1 2 3 4 5 6 7 8 42 24 6 17 30 13 90 47 2 {I =3}

0 1 2 3 4 5 6 7 8 42 24 6 17 30 13 90 47 2 {I =4}

0 1 2 3 4 5 6 7 8 42 24 6 17 30 13 90 47 2 {I =5}

0 1 2 3 4 5 6 7 8 42 24 6 17 30 13 90 47 2 {I =6}

0 1 2 3 4 5 6 7 8 42 24 6 17 30 13 90 47 2 {I =7}

It can be sort by insertion sort.

**B)** Selection sorting Methods

**C)** In this case, we use queue

D) Insertion Sort