# Week 1

1.Write a method to find the minimum number from an array of numbers

public static String findMinimum(int[] num)

{

int min = num[0];

for(int i=0;i<num.length;i++)

{

if(num[i]<min)

{

min = num[i];

}

}

return min;

}

# Week.2

1. Read the degree of two polynomials and their coefficients, all integers, from the standard input. The polynomial is of the form ( ) = ∗ + ⋯ + 1 ∗ 1 + , where 0 ≠ 0.
2. Write the pseudocode for adding two polynomials.

Step 1: Start

Step 2: function maximum(int x, int y)

{

Step 3: return (x > y) ? x : y;

}

Step 4: function addTwoPoly(int first[], int second[], int x, int y)

{

Step 5: int size = maximum(x, y);

Step 6: int sum[] = new int[size];

Step 7: for (int i = 0; i < x; i++)

{

Step 8: sum[i] = first[i];

}

Step 9: for (int i = 0; i < y; i++)

{

Step 10: sum[i] += second[i];

}

Step 11: return sum;

}

Step 12: function printPolynomials(int polynomia1l[], int y)

{

Step 13: for (int i = 0; i < y; i++)

{

Step 14: System.out.print(polynomial1[i]);

Step 15: if (i != 0) { Step 16: System.out.print ("x^" + i);

}

Step 17: if (i != n - 1)

{

Step 18: System.out.print (" + ");

}

}

}

Step 19: int first[] = {5, 0, 10, 6};

Step 20: int second[] = {1, 2, 4};

Step 21: int x = first.length;

Step 22: int y = second.length;

Step 23: System.out.println ("First polynomial is");

Step 24: printPolynomials (first, x);

Step 25: System.out.println (“Second polynomial is");

Step 26: printPolynomials (second, y);

Step 27: int sum[] = add(first, second, x, y);

Step 28: int size = maximum (x, y);

Step 29: System.out.println("sum polynomial is");

Step 30: printPoly(sum, size);

Step 31: Stop

4.Write the pseudocode and code for a function that determines the mean of elements in an array.

Step 1: int a[] new int {10, 100, 26, 33, 2}

Step 2: int size = a.length, sum = 0, avg

Step 3: for(int i = 0; i<a.length; i++)

{

Sum += a[i]

}

Step 4: avg = sum/size;

Step 5: display avg

Step 6: End

Function:

Public int calculate mean(int[] num, int size)

Sum

For(int I = 0; i<size; i++)

{

Sum += int[];

}

Return sum/size;

# Week.3

6.Write a recursive version of linear search on an array of integers. What is the time complexity of the algorithm? Use the BigO notation to express it.

     static int recSearch(int arr[], int l, int r, int x)

     {

          if (r < l)

             return -1;

          if (arr[l] == x)

             return l;

          if (arr[r] == x)

             return r;

          return recSearch(arr, l+1, r-1, x);

}

# Week.4

7.Implement a function to search an element in sequence using linear search.

Public bool linearSearch(int arr[], searchvalue)

{

For(int I = 0; i< arr[].length; i)

{

If(searchvalue == arr[i])

{

Return true;

}

}

Return false;

}

8. Write the pseudocode of binary search implementation.

Step 1: Start

Step 2: int arr[] = new int[] {1, 123, 77, 653, 123, 33}

Step 3: int size = arr.length

Step 4: int searchvalue

Step 5: print “Enter a search value”

Step 6: input searchvalue

Step 7: int left = 0, right = arr.length-1

Step 8: bool flag = false

Step 9: while (I <= r)

{

Mid = (l + (r-1))/2

If(arr[mid] == searchvalue)

{

Flag = true

}

If(arr[mid] .> searchvalue)

{

Right = mid -1

}

Else

{

Left = mid+1

}

}

If(flag == true)

{

System.out.print(“Element is present at the array”)

}

Else

{

System.out.print(“Element is not present in the array”)

}

Start 10: End

# Week 5

10. Write the pseudocode for enqueue() and dequeue() methods for queue.

Public string dequeue(){

if (isEmpty())

{

throw new

RuntimeException(“Queue underflow”);

}

else if (first==last)

{

String f= first item;

first= null;

last=null;

return f;

}

String f= first.item;

first= first.next;

return f;

Public void enqueue (String item)

{

Node x= new Node(item);

if (first==null && last== null)

{

first=x;

last=x;

return;//return back when first node is enqueued

}

last.next=x;

last=x;

}

Top of Form

Bottom of Form

# Week 6

11.Implement a function that inserts a node in a given binary tree.

public void insert(Node node,int value)

{

if(value<node.value)

{

if (node.left!=null)

{

insert(node.left,value);

}

else

{

System.out.println("inserted "+value +" to left of "+node.value);

node.left=new Node(value);

}

}

else if(value>node.value);

{

if (node.right!=null)

{

insert(node.right,value);

}

else

{

System.out.println("inserted "+value +" to right of "+node.value);

node.right=new Node (value);

}

}

}