1. (a) 0
2. (d) &&
3. (a) ~
4. (d) 25
5. (c) Double
6. (a) 66
7. (a) 24
8. (c) 40
9. (b) 26
10. (a) 10,20,30,40,50
11. (b) 11
12. (b) 1=1
13. (d) An object is a reference to an attribute
14. (b) 0
15. (c) Method is not defined properly
16. (a) Book(Book b){}

(b) It is the ability for a message/data to be processed in more than one form

1. (c) Polymorphism
2. (a) Method overloading
3. (c) Method overriding
4. (c) No Polymorphism
5. (d) As many as required
6. (a) Making atleast one member function as pure virtual function
7. class LargestSubArray {

// This function Prints the starting and ending

// indexes of the largest subarray with equal

// number of 0s and 1s. Also returns the size

// of such subarray.

int findSubArray(int arr[], int n)

{

int sum = 0;

int maxsize = -1, startindex = 0;

int endindex = 0;

// Pick a starting point as i

for (int i = 0; i < n - 1; i++) {

sum = (arr[i] == 0) ? -1 : 1;

// Consider all subarrays starting from i

for (int j = i + 1; j < n; j++) {

if (arr[j] == 0)

sum += -1;

else

sum += 1;

// If this is a 0 sum subarray, then

// compare it with maximum size subarray

// calculated so far

if (sum == 0 && maxsize < j - i + 1) {

maxsize = j - i + 1;

startindex = i;

}

}

}

endindex = startindex + maxsize - 1;

if (maxsize == -1)

System.out.println("No such subarray");

else

System.out.println(startindex + " to " + endindex);

return maxsize;

}

/\* Driver program to test the above functions \*/

public static void main(String[] args)

{

LargestSubArray sub;

sub = new LargestSubArray();

int arr[] = { 1, 0, 0, 1, 0, 1, 1 };

int size = arr.length;

sub.findSubArray(arr, size);

}

}

1. import java.util.Stack;

class Test

{

// method to get length of the longest valid

static int findMaxLen(String str)

{

int n = str.length();

// Create a stack and push -1

// as initial index to it.

Stack<Integer> stk = new Stack<>();

stk.push(-1);

// Initialize result

int result = 0;

// Traverse all characters of given string

for (int i = 0; i < n; i++)

{

// If opening bracket, push index of it

if (str.charAt(i) == '(')

stk.push(i);

// // If closing bracket, i.e.,str[i] = ')'

else

{

// Pop the previous

// opening bracket's index

if(!stk.empty())

stk.pop();

// Check if this length

// formed with base of

// current valid substring

// is more than max

// so far

if (!stk.empty())

result

= Math.max(result,

i - stk.peek());

// If stack is empty. push

// current index as base

// for next valid substring (if any)

else

stk.push(i);

}

}

return result;

}

// Driver code

public static void main(String[] args)

{

String str = "((()()";

// Function call

System.out.println(findMaxLen(str));

str = "()(()))))";

// Function call

System.out.println(findMaxLen(str));

}

}