FOP Mock Answer Sheet

If you have any questions/unclear why the answer is so, it is recommended that you try debug it by coding it out. If you're still stuck, please feel free to join our discord and ask for help!

discord.gg/DBSbU4p4zC

1. C
2. B
3. C
4. B
5. A
6. D
7. D
8. C
9. D
10. D
11. A
12. C
13. A
14. C
15. B
16. D
17. B
18. C
19. B
20. A
21. B
22. D
23. D
24. D
25. C
26. C
27. D
28. A
29. A

30.A
31.B
32.A

Explanation:

The value of j starts at 0. Then the condition (j++ < 5) is checked, so because j(which is 0), is less than 5, we enter the for loop. Notice that the condition is j++ and therefore when **entering the for loop for the first time, j already has a value of 1**. At the end of the for loop, j is incremented by j++ to 2. The condition (j++ < 5) is checked again, **and such j enters the for loop with a value of 3.** This happens once more, we enter the for loop with j having a value of 5. At the end of this for loop, j++ increases j to 6. Lastly, the condition(j++ < 5) is checked, as 6 is not less than 5, we get out of the for loop. However the j++ still has an effect on j, increasing j once more to a value of 7. As j is initialized with a var, our console log prints 7 instead of throwing an error.

33.A

Explanation:
Starting off, x(which is 0) and y(which is 1) fulfills the condition x + y < 5. Therefore we enter the loop. Inside the loop, x * y = 0 * 1 = 0. Thus **maximum** does not change. After every for loop, the right hand side (x++, y+= 2) tells us x will increase by 1, while y will increase by 2. x becomes 1 and y becomes 3. When we enter the for loop, **maximum is now set to 1 * 3 = 3.** This is the last loop as when we increase x to 2 and y to 5, the condition x + y < 5 breaks. As a result, **the value we print out is 3**.

34.C

Explanation:
If we look at the condition, the for loop breaks as soon as the index is outside the range of the array. The loop starts off at i = 1, and adds the value at index 1(which is a value of 2) to the sum. Next, the code **i += i + 2** tells us that i **increases by i + 2**. To make it more simple, it is better to **think of this as setting i to i + i + 2**, or to simplify it further, i = 2 * i + 2. If we plug the current value of i in this equation, we can see that i is set to (2 * 1) + 2 = 4. Next, due to the right hand side of the for loop, i increases by 1. As i = 5 satisfies the condition of

being less than array.length(which is 9), we add the array value at index 5(array value is 6) to the sum. We set i to its new value 2 * (5) + 2, which is 12. Here, as i already breaks the condition in the for loop and because we are already on the last line of the loop, we no longer need to continue anymore. As a result, the sum is 2 + 6 = 8.

35.A

Explanation:

Note that the variable **out** is initialized as a string. When we enter the for loop, we see that x is going to loop through 0,1,2 and 3. Starting off with 0, there is no case for 0, as a result we add 4 to out. **out becomes "4"**. Next, there is a case for 1, however there is no break statement. In this case, we keep going down the switch case until there is a break statement. This means we'll do both out += 1 and out += 2, until breaking after out += 2. **out becomes "412**". Next, there is a case for 2, which is just out += 2 and breaking immediately after. **out becomes "4122".** The last number 3, also has no break statement. As a result, we continue going down, performing both out += 3 and out += 4 to the variable out. **out becomes "412234".** The for loop, ends and thus we print out our result.

PART 2

1a)

```
secret_num.push(Math.floor(Math.random( ) * 6)+1)
```

1b)

```
var input = require('readline-sync')

userinput = []
for(var i = 1;i <= 4;i++){
  do{
    var current_num= input.questionInt("enter number "+i+"
: ")
    if(current_num<1 || current_num>6){
      console.log("you have entered the number wrongly")
    }
  }while(current_num<1 || current_num>6)
  userinput.push(current_num)

}
console.log(userinput)
```

1c)

```
no_correct = 0
for(var i = 0 ;i<4;i++){
  if(secret_num[i] == userinput[i]){
    no_correct +=1
  }
}
console.log("number of correct is "+no_correct)
```

2)

```
//max is to store the maximum cookies and cake that can be bought
max = 0
for(var i = 0 ; i< cookie.length;i++){
  for(var f = 0 ; f< cake.length;f++){
    if ((cookie[i]+cake[f])<=60 && (cookie[i]+cake[f])>max){
      max = cookie[i]+cake[f]
    }
  }
}

if(max == 0 ){
  console.log("error")
}else{
  console.log(max)
}
```

Explanation :

1a) go read the fop slides

1b)i) the if statement needs to make sure variable current_num is between 1 to 6

1b)ii) the while loop will continue if current_num is not 1 to 6, just a trick question to make you doubt your answer.

1c) **The idea of the program is to check how many elements will be equal at the same index for list** secret_num and userinput **.**

E.g . secret_num = [1,2,3,4] and userinput = [1,2,4,6]

“Number of correct is 2” will be displayed

This is because secret_num[0] is equal to userinput[0] AND secret_num[1] is also equal to userinput[1] .So there’s 2 index that selects the element that is equal.Therefore the answer for this example is 2.

2)
Explanation :

The logic of the program is to use nested for loops to find out all the combinations of bundles ivan can buy .

2i) this condition checks if the combination exceeds the total amount of food Ivan can eat.
2ii) Since the objective of this code is to find out what is the maximum cookies and cake that can be bought , we will need to update Variable **max** to store the larger total combination of cakes and cookies
2iii) if Variable **max** is still equals to 0 which is the default value in the beginning of the program , no combination have met condition 2i) and 2ii) .