



Assignment Solutions : Backtracking Assignment - 1

Q1 Given n as input. Generate all strings that are palindromes with the number of digits as 'n'.

For example a palindrome of size 3 can be 313, 121, 030.

Note it can even contain leading zeros

Input : n = 2

Output : 00, 11, 22, 33, 44, 55, 66, 77, 88, 99

Code link: <https://pastebin.com/it6rfLux>

Explanation :

- Here we do a recursive function call, as the number is palindrome the second half will be the reverse of the first half. Then we only need to solve for the first half of the string and can simply reverse for the second half.
- We can do that using recursion. At each step just take the current string and keep on adding the elements. And when we revert back to the function call, just pop back that element.
- Keep on doing this till we reach half of the string and then simply calculate and print the answer.

Output:

```
2
00, 11, 22, 33, 44, 55, 66, 77, 88, 99,
...Program finished with exit code 0
Press ENTER to exit console.
```

Q2 Check if the product of some subset of an array is equal to the target value.

Where n is the size of the input array.

Note: Each index value can be used only once.

Input : n = 5 , target = 16

Array = [2 3 2 5 4]

Here the target will be equal to $2 \times 2 \times 4 = 16$

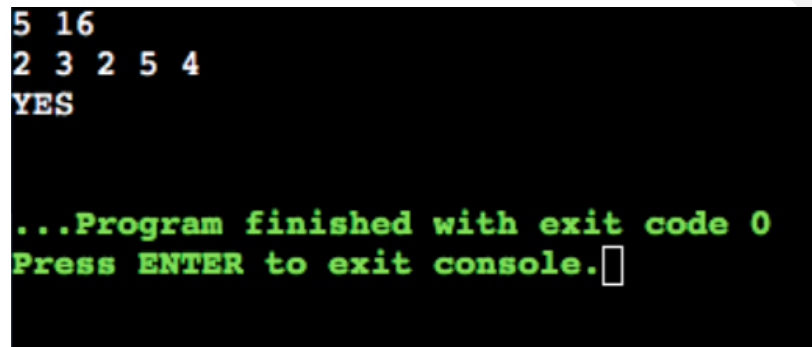
Output : YES

Code link: <https://pastebin.com/nKckmzYB>

Explanation :

- Just make a recursive function and take two cases. In the first case we consider taking the element for product and multiply it with product
- In the second case, we do not take the element and leave that element.
- This way we reach the point which is the end of the array, simply check whether the product is equal to the target.

Output:



```
5 16
2 3 2 5 4
YES

...Program finished with exit code 0
Press ENTER to exit console.
```

Q3 Given an integer array nums that may contain duplicates, return all possible subsets(the power set).

The solution set must not contain duplicate subsets. Return the solution in any order.

Sample Input: nums=[1,1,2]

Sample Output: [],[1],[1,2],[1,1],[1,1,2],[2]

Sample Input: nums=[1,2]

Sample Output: [],[1],[2],[1,2]

Code link: <https://pastebin.com/wgvJPj51>

Explanation :

- Sort the input array to make it easier to identify duplicates.
- Define a recursive function that takes as input the current subset, the index of the next element to consider, and the set of all subsets found so far. The function should do the following:
 - a. Add the current subset to the set of subsets found so far.
 - b. Loop over the remaining elements in the input array starting from the next index. For each element:
 - i. If the element is a duplicate of the previous element (i.e., `nums[i] == nums[i-1]`), continue to the next element to avoid duplicate subsets.
 - ii. Add the current element to the current subset.
 - iii. Recursively call the function with the updated subset and the next index.
 - iv. Remove the current element from the current subset to prepare for the next iteration of the loop.
- Call the recursive function with an empty subset and the starting index of 0.
- Convert the set of subsets found so far to a list and return it.

Output:

```
1
1 2
1 2 2
2
2 2

...Program finished with exit code 0
Press ENTER to exit console.
```

Q4 Given a string `s`, you can transform every letter individually to be lowercase or uppercase to create another string.

Return a list of all possible strings we could create. Return the output in any order.

Sample Input: `s="a1"`

Sample Output : `["a1","A1"]`

Sample Input: `s="bc12"`

Sample Output: `["bc12","bC12","Bc12","BC12"]`

Code link: <https://pastebin.com/CQsGk9S3>

Explanation :

1. Create an empty result list to store all possible strings.
2. Start a recursive function to generate all possible combinations of lowercase and uppercase letters for each character in the input string.
3. In the recursive function, iterate through each character in the input string and generate two possibilities for each character – either convert it to lowercase or uppercase.
4. Append the lowercase or uppercase character to the current result string and call the recursive function again with the remaining substring and the updated result string.
5. When we reach the end of the input string, add the current result string to the result list.
6. Return the result list.

Output:

```
Enter the string : bc12
bc12
bC12
Bc12
BC12

...Program finished with exit code 0
Press ENTER to exit console.□
```

