1. Write a program in the following steps
    a. Generates 10 Random 3 Digit number.
    b. Store this random numbers into a array.
    c. Then find the 2nd largest and the 2nd smallest element without sorting the array.

```
  GNU nano 6.4                    random3number2ndSmallLarge.sh
ran_array=()
for ((i=0; i<10; i++)); do
  ran_array[i]=$((RANDOM % 900 + 100))
done

echo "Random numbers: ${ran_array[*]}"



smallest=${ran_array[0]}
s_smallest=1000
largest=0
s_largest=0



for number in "${ran_array[@]}"; do
  if [[ $number -lt $smallest ]]; then
    s_smallest=$smallest
    smallest=$number
  elif [[ $number -lt $s_smallest && $number -ne $smallest ]]; then
    s_smallest=$number
  fi

  if [[ $number -gt $largest ]]; then
    s_largest=$largest
    largest=$number
  elif [[ $number -gt $s_largest && $number -ne $largest ]]; then
    s_largest=$number
  fi
done


echo "second smallest number: $s_smallest"
echo "second largest number: $s_largest"
```

```
Shri@PRODUCTIVITY-4 MINGW64 ~/Testing_Bridge/repoPortal/repo1/D7 (main)
$ sh random3number2ndSmallLarge.sh
Random numbers: 166 484 988 628 690 102 410 722 447 723
second smallest number:\t166
second largest number:\t723

Shri@PRODUCTIVITY-4 MINGW64 ~/Testing_Bridge/repoPortal/repo1/D7 (main)
$ sh random3number2ndSmallLarge.sh
Random numbers: 758 448 804 251 286 143 712 608 952 252
second smallest number: 251
second largest number: 804
```

2. Extend the above program to sort the array and then find the 2nd largest and the 2nd smallest element.

```
  GNU nano 6.4                          2_rn3num2ndslSORT.sh
rndN=()

for i in {1..10}; do
  rndN+=($((RANDOM % 900 + 100)))
done

echo "Random numbers: ${rndN[*]}"

srtN=($(printf "%d\n" "${rndN[@]}" | sort -n))

echo "Sorted numbers: ${srtN[*]}"

s_smallest=${srtN[1]}
s_largest=${srtN[-2]}

echo "second Smallest: $s_smallest"
echo "second Largest: $s_largest"
```

```
Shri@PRODUCTIVITY-4 MINGW64 ~/Testing_Bridge/repoPortal/repo1/D7 (main)
$ sh 2_rn3num2ndslSORT.sh
Random numbers: 152 220 221 204 959 405 790 478 516 670
Sorted numbers: 152 204 220 221 405 478 516 670 790 959
second Smallest: 204
second Largest: 790

Shri@PRODUCTIVITY-4 MINGW64 ~/Testing_Bridge/repoPortal/repo1/D7 (main)
$ sh 2_rn3num2ndslSORT.sh
Random numbers: 552 453 859 302 709 116 541 836 929 566
Sorted numbers: 116 302 453 541 552 566 709 836 859 929
second Smallest: 302
second Largest: 859
```

3. *Extend the Prime Factorization Program to store all the Prime Factors of a number n into an array and finally display the output.*

   -(Lowest Common Multiplier)

```
  GNU nano 6.4                                3_primFact.sh
is_prime() {
    num=$1
    if [ $num -le 1 ]; then
        echo 0
        return
    fi

    for ((i=2; i*i<=num; i++)); do
        if [ $((num % i)) -eq 0 ]; then
            echo 0
            return
        fi
    done
    echo 1
}

read -p "Enter a number: " n

primeFact=()

while [ $(is_prime $n) -eq 0 ]; do
    for ((i=2; i<=n; i++)); do
        if [ $((n % i)) -eq 0 ] && [ $(is_prime $i) -eq 1 ]; then
            primeFact+=($i)
            n=$((n / i))
            break
        fi
    done
done

if [ $n -gt 1 ]; then
    primeFact+=($n)
fi

echo "Prime factors: ${primeFact[@]}"
```

```
Shri@PRODUCTIVITY-4 MINGW64 ~/Testing_Bridge/repoPortal/repo1/D7 (main)
$ sh 3_primFact.sh
Enter a number: 56
Prime factors: 2 2 2 7

Shri@PRODUCTIVITY-4 MINGW64 ~/Testing_Bridge/repoPortal/repo1/D7 (main)
$ nano 3_primFact.sh

Shri@PRODUCTIVITY-4 MINGW64 ~/Testing_Bridge/repoPortal/repo1/D7 (main)
$ sh 3_primFact.sh
Enter a number: 1000
Prime factors: 2 2 2 5 5 5

Shri@PRODUCTIVITY-4 MINGW64 ~/Testing_Bridge/repoPortal/repo1/D7 (main)
$ sh 3_primFact.sh
Enter a number: 34
Prime factors: 2 17

Shri@PRODUCTIVITY-4 MINGW64 ~/Testing_Bridge/repoPortal/repo1/D7 (main)
$ sh 3_primFact.sh 63
Enter a number: 63
Prime factors: 3 3 7
```

4. *Write a Program to show Sum of three Integer adds to ZERO*

```
  GNU nano 6.4                                4_tripplets.sh
read -p "Enter the array of integers (space-separated): " -a arr

n=${#arr[@]}

found=0

for ((i=0; i<n-2; i++)); do
    for ((j=i+1; j<n-1; j++)); do
        for ((k=j+1; k<n; k++)); do
            if [ $((arr[i] + arr[j] + arr[k])) -eq 0 ]; then
                echo "Triplet: ${arr[i]} ${arr[j]} ${arr[k]}"
                found=1
            fi
        done
    done
done

if [ $found -eq 0 ]; then
    echo "No triplet found"
fi
```

```
Shri@PRODUCTIVITY-4 MINGW64 ~/Testing_Bridge/repoPortal/repo1/D7 (main)
$ sh 4_tripplets.sh
Enter the array of integers (space-separated):  4  7 4 32 4 -5 -2 -45 -0 0 -1 1 0
Triplet: 4 -5 1
Triplet: 7 -5 -2
Triplet: 4 -5 1
Triplet: 4 -5 1
Triplet: -0 0 0
Triplet: -0 -1 1
Triplet: 0 -1 1
Triplet: -1 1 0
```

5. Take a range from 0 – 100, find the digits that are repeated twice like 33, 77, etc and store them in an array

```
  GNU nano 6.4                                5_repeatedDigits.sh
repeat_digits=()

for ((i=10; i<=100; i++)); do
  first_digit=$((i % 10))
  second_digit=$((i / 10))
  if [ $first_digit -eq $second_digit ]; then
    repeat_digits+=($i)
  fi
done

echo "Repeated digit numbers: ${repeat_digits[@]}"
```

```
Shri@PRODUCTIVITY-4 MINGW64 ~/Testing_Bridge/repoPortal/repo1/D7 (main)
$ sh 5_repeatedDigits.sh
Repeated digit numbers: 11 22 33 44 55 66 77 88 99
```