Day 7 Practice Problems

Solved by: SAPTAK DAS(saptakds@gmail.com)

Arrays Practice Problems

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.

```
#!/bin/bash -x
arr=()
for i in `seq 0 9`
do
   randomNo=$((100+$RANDOM%999))
   arr[i]=$randomNo
done
first=${arr[0]}
min=$first
minTwo=$first
max=$first
maxTwo=$first
for j in ${arr[@]}
do
   if [ $j -lt $min ]
   then
      minTwo=$min
      min=$j
   elif [ $j -lt $minTwo -a $j -ne $min ]
   then
      minTwo=$j
   fi
   if [ $j -gt $max ]
   then
      maxTwo=$max
      max=$j
   elif [ $j -gt $maxTwo -a $j -ne $max ]
   then
      maxTwo=$j
   fi
done
echo "Second minimum: $minTwo"
echo "Second maximum: $maxTwo"
```

```
+ arr=()
++ seq 0 9
+ for i in `seq 0 9`
+ randomNo=880
+ arr[i]=880
+ for i in `seq 0 9`
+ randomNo=105
+ arr[i]=105
+ for i in `seq 0 9`
+ randomNo=160
+ arr[i]=160
+ for i in `seq 0 9`
+ randomNo=365
+ arr[i]=365
+ for i in `seq 0 9`
+ randomNo=727
+ arr[i]=727
+ for i in `seq 0 9`
+ randomNo=416
+ arr[i]=416
+ for i in `seq 0 9`
+ randomNo=392
+ arr[i]=392
+ for i in `seq 0 9`
+ randomNo=269
+ arr[i]=269
+ for i in `seq 0 9`
+ randomNo=253
+ arr[i]=253
+ for i in `seq 0 9`
+ randomNo=490
+ arr[i]=490
+ first=880
+ min=880
+ minTwo=880
+ max=880
+ maxTwo=880
+ for j in ${arr[@]}
+ '[' 880 -lt 880 ']'
+ '[' 880 -lt 880 -a 880 -ne 880 ']'
+ '[' 880 -gt 880 ']'
+ '[' 880 -gt 880 -a 880 -ne 880 ']'
+ for j in ${arr[@]}
+ '[' 105 -lt 880 ']'
+ minTwo=880
+ min=105
+ '[' 105 -gt 880 ']'
+ '[' 105 -gt 880 -a 105 -ne 880 ']'
+ for j in ${arr[@]}
+ '[' 160 -lt 105 ']'
+ '[' 160 -lt 880 -a 160 -ne 105 ']'
+ minTwo=160
+ '[' 160 -gt 880 ']'
+ '[' 160 -gt 880 -a 160 -ne 880 ']'
+ for j in ${arr[@]}
+ '[' 365 -lt 105 ']'
+ '[' 365 -lt 160 -a 365 -ne 105 ']'
+ '[' 365 -gt 880 ']'
+ '[' 365 -gt 880 -a 365 -ne 880 ']'
+ for j in ${arr[@]}
+ '[' 727 -lt 105 ']'
+ '[' 727 -lt 160 -a 727 -ne 105 ']'
+ '[' 727 -gt 880 ']'
+ '[' 727 -gt 880 -a 727 -ne 880 ']'
+ for j in ${arr[@]}
+ '[' 416 -lt 105 ']'
+ '[' 416 -lt 160 -a 416 -ne 105 ']'
+ '[' 416 -gt 880 ']'
+ '[' 416 -gt 880 -a 416 -ne 880 ']'
+ for j in ${arr[@]}
+ '[' 392 -lt 105 ']'
+ '[' 392 -lt 160 -a 392 -ne 105 ']'
+ '[' 392 -gt 880 ']'
+ '[' 392 -gt 880 -a 392 -ne 880 ']'
+ for j in ${arr[@]}
+ '[' 269 -lt 105 ']'
+ '[' 269 -lt 160 -a 269 -ne 105 ']'
+ '[' 269 -gt 880 ']'
+ '[' 269 -gt 880 -a 269 -ne 880 ']'
+ for j in ${arr[@]}
+ '[' 253 -lt 105 ']'
+ '[' 253 -lt 160 -a 253 -ne 105 ']'
+ '[' 253 -gt 880 ']'
+ '[' 253 -gt 880 -a 253 -ne 880 ']'
+ for j in ${arr[@]}
+ '[' 490 -lt 105 ']'
+ '[' 490 -lt 160 -a 490 -ne 105 ']'
+ '[' 490 -gt 880 ']'
+ '[' 490 -gt 880 -a 490 -ne 880 ']'
+ echo 'Second minimum: 160'
Second minimum: 160
+ echo 'Second maximum: 880'
Second maximum: 880
```

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

```
#!/bin/bash -x
function bubbleSort(){
   ar=($@)
   arLen=${#ar[@]}
   for(( i=0 ; i<$arLen ; i++))
   do
      for(( j=0 ; j<$arLen-$i-1 ; j++ ))
      do
         if [ ${ar[j]} -gt ${ar[$((j+1))]} ]
         then
            temp=${ar[j]}
            ar[$j]=${ar[$((j+1))]}
            ar[$((j+1))]=$temp
         fi
      done
   done
   echo ${ar[@]}
}
arr=()
for k in `seq 0 9`
do
   randomNo=$((100+$RANDOM%999))
   arr[k]=$randomNo
done
sortedArr=(`bubbleSort ${arr[@]}`)
arrLen=${#sortedArr[@]}
minTwo=${sortedArr[1]}
maxTwo=${sortedArr[$arrLen-2]}
echo "Second minimum: $minTwo"
echo "Second maximum: $maxTwo"
```

```
+ arr=()
++ seq 0 9
+ for k in `seq 0 9`
+ randomNo=983
+ arr[k]=983
+ for k in `seq 0 9`
+ randomNo=968
+ arr[k]=968
+ for k in `seq 0 9`
+ randomNo=469
+ arr[k]=469
+ for k in `seq 0 9`
+ randomNo=876
+ arr[k]=876
+ for k in `seq 0 9`
+ randomNo=841
+ arr[k]=841
+ for k in `seq 0 9`
+ randomNo=483
+ arr[k]=483
+ for k in `seq 0 9`
+ randomNo=795
+ arr[k]=795
+ for k in `seq 0 9`
+ randomNo=846
+ arr[k]=846
+ for k in `seq 0 9`
+ randomNo=423
+ arr[k]=423
+ for k in `seq 0 9`
+ randomNo=852
+ arr[k]=852
+ sortedArr=(`bubbleSort ${arr[@]}`)
++ bubbleSort 983 968 469 876 841
483 795 846 423 852
++ ar=($@)
++ arLen=10
++ (( i=0  ))
++ (( i<10  ))
++ (( j=0  ))
++ (( j<10-0-1  ))
++ '[' 983 -gt 968 ']'
++ temp=983
++ ar[$j]=968
++ ar[$((j+1))]=983
++ (( j++  ))
++ (( j<10-0-1  ))
++ '[' 983 -gt 469 ']'
++ temp=983
++ ar[$j]=469
++ ar[$((j+1))]=983
++ (( j++  ))
++ (( j<10-0-1  ))
++ '[' 983 -gt 876 ']'
++ temp=983
++ ar[$j]=876
++ ar[$((j+1))]=983
++ (( j++  ))
++ (( j<10-0-1  ))
++ '[' 983 -gt 841 ']'
++ temp=983
++ ar[$j]=841
++ ar[$((j+1))]=983
++ (( j++  ))
++ (( j<10-0-1  ))
++ '[' 983 -gt 483 ']'
++ temp=983
++ ar[$j]=483
++ ar[$((j+1))]=983
++ (( j++  ))
++ (( j<10-0-1  ))
++ '[' 983 -gt 795 ']'
++ temp=983
++ ar[$j]=795
++ ar[$((j+1))]=983
++ (( j++  ))
++ (( j<10-0-1  ))
++ '[' 983 -gt 846 ']'
++ temp=983
++ ar[$j]=846
++ ar[$((j+1))]=983
++ (( j++  ))
++ (( j<10-0-1  ))
++ '[' 983 -gt 423 ']'
++ temp=983
++ ar[$j]=423
++ ar[$((j+1))]=983
++ (( j++  ))
++ (( j<10-0-1  ))
++ '[' 983 -gt 852 ']'
++ temp=983
++ ar[$j]=852
++ ar[$((j+1))]=983
++ (( j++  ))
++ (( j<10-0-1  ))
++ (( i++ ))
++ (( i<10  ))
++ (( j=0  ))
++ (( j<10-1-1  ))
++ '[' 968 -gt 469 ']'
++ temp=968
++ ar[$j]=469
++ ar[$((j+1))]=968
++ (( j++  ))
++ (( j<10-1-1  ))
++ '[' 968 -gt 876 ']'
++ temp=968
++ ar[$j]=876
++ ar[$((j+1))]=968
++ (( j++  ))
++ (( j<10-1-1  ))
++ '[' 968 -gt 841 ']'
++ temp=968
++ ar[$j]=841
++ ar[$((j+1))]=968
++ (( j++  ))
++ (( j<10-1-1  ))
++ '[' 968 -gt 483 ']'
++ temp=968
++ ar[$j]=483
++ ar[$((j+1))]=968
++ (( j++  ))
++ (( j<10-1-1  ))
++ '[' 968 -gt 795 ']'
++ temp=968
++ ar[$j]=795
++ ar[$((j+1))]=968
++ (( j++  ))
++ (( j<10-1-1  ))
++ '[' 968 -gt 846 ']'
++ temp=968
++ ar[$j]=846
++ ar[$((j+1))]=968
++ (( j++  ))
++ (( j<10-1-1  ))
++ '[' 968 -gt 423 ']'
++ temp=968
++ ar[$j]=423
++ ar[$((j+1))]=968
++ (( j++  ))
++ (( j<10-1-1  ))
++ '[' 968 -gt 852 ']'
++ temp=968
++ ar[$j]=852
++ ar[$((j+1))]=968
++ (( j++  ))
++ (( j<10-1-1  ))
++ (( i++ ))
++ (( i<10  ))
++ (( j=0  ))
++ (( j<10-2-1  ))
++ '[' 469 -gt 876 ']'
++ (( j++  ))
++ (( j<10-2-1  ))
++ '[' 876 -gt 841 ']'
++ temp=876
++ ar[$j]=841
++ ar[$((j+1))]=876
++ (( j++  ))
++ (( j<10-2-1  ))
++ '[' 876 -gt 483 ']'
++ temp=876
++ ar[$j]=483
++ ar[$((j+1))]=876
++ (( j++  ))
++ (( j<10-2-1  ))
++ '[' 876 -gt 795 ']'
++ temp=876
++ ar[$j]=795
++ ar[$((j+1))]=876
++ (( j++  ))
++ (( j<10-2-1  ))
++ '[' 876 -gt 846 ']'
++ temp=876
++ ar[$j]=846
++ ar[$((j+1))]=876
++ (( j++  ))
++ (( j<10-2-1  ))
++ '[' 876 -gt 423 ']'
++ temp=876
++ ar[$j]=423
++ ar[$((j+1))]=876
++ (( j++  ))
++ (( j<10-2-1  ))
++ '[' 876 -gt 852 ']'
++ temp=876
++ ar[$j]=852
++ ar[$((j+1))]=876
++ (( j++  ))
++ (( j<10-2-1  ))
++ (( i++ ))
++ (( i<10  ))
++ (( j=0  ))
++ (( j<10-3-1  ))
++ '[' 469 -gt 841 ']'
++ (( j++  ))
++ (( j<10-3-1  ))
++ '[' 841 -gt 483 ']'
++ temp=841
++ ar[$j]=483
++ ar[$((j+1))]=841
++ (( j++  ))
++ (( j<10-3-1  ))
++ '[' 841 -gt 795 ']'
++ temp=841
++ ar[$j]=795
++ ar[$((j+1))]=841
++ (( j++  ))
++ (( j<10-3-1  ))
++ '[' 841 -gt 846 ']'
++ (( j++  ))
++ (( j<10-3-1  ))
++ '[' 846 -gt 423 ']'
++ temp=846
++ ar[$j]=423
++ ar[$((j+1))]=846
++ (( j++  ))
++ (( j<10-3-1  ))
++ '[' 846 -gt 852 ']'
++ (( j++  ))
++ (( j<10-3-1  ))
++ (( i++ ))
++ (( i<10  ))
++ (( j=0  ))
++ (( j<10-4-1  ))
++ '[' 469 -gt 483 ']'
++ (( j++  ))
++ (( j<10-4-1  ))
++ '[' 483 -gt 795 ']'
++ (( j++  ))
++ (( j<10-4-1  ))
++ '[' 795 -gt 841 ']'
++ (( j++  ))
++ (( j<10-4-1  ))
++ '[' 841 -gt 423 ']'
++ temp=841
++ ar[$j]=423
++ ar[$((j+1))]=841
++ (( j++  ))
++ (( j<10-4-1  ))
++ '[' 841 -gt 846 ']'
++ (( j++  ))
++ (( j<10-4-1  ))
++ (( i++ ))
++ (( i<10  ))
++ (( j=0  ))
++ (( j<10-5-1  ))
++ '[' 469 -gt 483 ']'
++ (( j++  ))
++ (( j<10-5-1  ))
++ '[' 483 -gt 795 ']'
++ (( j++  ))
++ (( j<10-5-1  ))
++ '[' 795 -gt 423 ']'
++ temp=795
++ ar[$j]=423
++ ar[$((j+1))]=795
++ (( j++  ))
++ (( j<10-5-1  ))
++ '[' 795 -gt 841 ']'
++ (( j++  ))
++ (( j<10-5-1  ))
++ (( i++ ))
++ (( i<10  ))
++ (( j=0  ))
++ (( j<10-6-1  ))
++ '[' 469 -gt 483 ']'
++ (( j++  ))
++ (( j<10-6-1  ))
++ '[' 483 -gt 423 ']'
++ temp=483
++ ar[$j]=423
++ ar[$((j+1))]=483
++ (( j++  ))
++ (( j<10-6-1  ))
++ '[' 483 -gt 795 ']'
++ (( j++  ))
++ (( j<10-6-1  ))
++ (( i++ ))
++ (( i<10  ))
++ (( j=0  ))
++ (( j<10-7-1  ))
++ '[' 469 -gt 423 ']'
++ temp=469
++ ar[$j]=423
++ ar[$((j+1))]=469
++ (( j++  ))
++ (( j<10-7-1  ))
++ '[' 469 -gt 483 ']'
++ (( j++  ))
++ (( j<10-7-1  ))
++ (( i++ ))
++ (( i<10  ))
++ (( j=0  ))
++ (( j<10-8-1  ))
++ '[' 423 -gt 469 ']'
++ (( j++  ))
++ (( j<10-8-1  ))
++ (( i++ ))
++ (( i<10  ))
++ (( j=0  ))
++ (( j<10-9-1  ))
++ (( i++ ))
++ (( i<10  ))
++ echo 423 469 483 795 841 846
852 876 968 983
+ arrLen=10
+ minTwo=469
+ maxTwo=968
+ echo 'Second minimum: 469'
Second minimum: 469
+ echo 'Second maximum: 968'
Second maximum: 968
```

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

```
#!/bin/bash -x
function isPrime(){
    if [ $1 -eq 2 ]
    then
        return 0
    elif [ $(($1 % 2)) -eq 0 ]
    then
        return 1
    fi
    for(( i=3 ; i<=$(($1/2)) ; i+=2 ))
    do
        if [ $(($1 % $i)) -eq 0 ]
        then
            return 1
        fi
    done
    return 0
}
arr=()
k=0
read -p "Enter N: " N
for (( j=2 ; $((j*j))<=$N ; j++ ))
do
if [ $(($N % $j)) -eq 0 ]
    then
        if isPrime $j
        then
            arr[$k]=$j
            ((k++))
        else
            continue
        fi
    fi
done
echo ${arr[@]}
```

```
+ arr=()
+ k=0
+ read -p 'Enter N: ' N
Enter N: 30
+ (( j=2  ))
+ (( 4<=30  ))
+ '[' 0 -eq 0 ']'
+ isPrime 2
+ '[' 2 -eq 2 ']'
+ return 0
+ arr[$k]=2
+ (( k++ ))
+ (( j++  ))
+ (( 9<=30  ))
+ '[' 0 -eq 0 ']'
+ isPrime 3
+ '[' 3 -eq 2 ']'
+ '[' 1 -eq 0 ']'
+ (( i=3  ))
+ (( i<=1  ))
+ return 0
+ arr[$k]=3
+ (( k++ ))
+ (( j++  ))
+ (( 16<=30  ))
+ '[' 2 -eq 0 ']'
+ (( j++  ))
+ (( 25<=30  ))
+ '[' 0 -eq 0 ']'
+ isPrime 5
+ '[' 5 -eq 2 ']'
+ '[' 1 -eq 0 ']'
+ (( i=3  ))
+ (( i<=2  ))
+ return 0
+ arr[$k]=5
+ (( k++ ))
+ (( j++  ))
+ (( 36<=30  ))
+ echo 2 3 5
2 3 5
```

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

```
#!/bin/bash -x
read -p "Enter no. of integers:" n
arr=()
for num in `seq 0 $(($n-1))`
do
   read -p "Enter no.:" arr[$num]
done
flag=0
for (( i=0 ; i<$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[$k]})) -eq 0 ]
         then
            echo "${arr[$i]} ${arr[$j]} ${arr[$k]}"
            flag=1
         fi
      done
   done
done
if [ $flag -eq 0 ]
then
   echo "Doesn't exist"
fi
```

```
+ read -p 'Enter no. of integers:' n
Enter no. of integers:5
+ arr=()
++ seq 0 4
+ for num in `seq 0 $(($n-1))`
+ read -p 'Enter no.:' 'arr[0]'
Enter no.:0
+ for num in `seq 0 $(($n-1))`
+ read -p 'Enter no.:' 'arr[1]'
Enter no.:-1
+ for num in `seq 0 $(($n-1))`
+ read -p 'Enter no.:' 'arr[2]'
Enter no.:2
+ for num in `seq 0 $(($n-1))`
+ read -p 'Enter no.:' 'arr[3]'
Enter no.:-3
+ for num in `seq 0 $(($n-1))`
+ read -p 'Enter no.:' 'arr[4]'
Enter no.:1
+ flag=0
+ (( i=0  ))
+ (( i<5-2  ))
+ (( j=0+1 ))
+ (( j<5-1  ))
+ (( k=1+1  ))
+ (( k<5  ))
+ '[' 1 -eq 0 ']'
+ (( k++  ))
+ (( k<5  ))
+ '[' -4 -eq 0 ']'
+ (( k++  ))
+ (( k<5  ))
+ '[' 0 -eq 0 ']'
+ echo '0 -1 1'
0 -1 1
+ flag=1
+ (( k++  ))
+ (( k<5  ))
+ (( j++  ))
+ (( j<5-1  ))
+ (( k=2+1  ))
+ (( k<5  ))
+ '[' -1 -eq 0 ']'
+ (( k++  ))
+ (( k<5  ))
+ '[' 3 -eq 0 ']'
+ (( k++  ))
+ (( k<5  ))
+ (( j++  ))
+ (( j<5-1  ))
+ (( k=3+1  ))
+ (( k<5  ))
+ '[' -2 -eq 0 ']'
+ (( k++  ))
+ (( k<5  ))
+ (( j++  ))
+ (( j<5-1  ))
+ (( i++  ))
+ (( i<5-2  ))
+ (( j=1+1 ))
+ (( j<5-1  ))
+ (( k=2+1  ))
```

```
+ (( k<5  ))
+ '[' -2 -eq 0 ']'
+ (( k++  ))
+ (( k<5  ))
+ '[' 2 -eq 0 ']'
+ (( k++  ))
+ (( k<5  ))
+ (( j++  ))
+ (( j<5-1  ))
+ (( k=3+1  ))
+ (( k<5  ))
+ '[' -3 -eq 0 ']'
+ (( k++  ))
+ (( k<5  ))
+ (( j++  ))
+ (( j<5-1  ))
+ (( i++  ))
+ (( i<5-2  ))
+ (( j=2+1 ))
+ (( j<5-1  ))
+ (( k=3+1  ))
+ (( k<5  ))
+ '[' 0 -eq 0 ']'
+ echo '2 -3 1'
2 -3 1
+ flag=1
+ (( k++  ))
+ (( k<5  ))
+ (( j++  ))
+ (( j<5-1  ))
+ (( i++  ))
+ (( i<5-2  ))
+ '[' 1 -eq 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

```
#!/bin/bash -x
function isPalindrome(){
   num=$1
   sum=0
   while [ $num -ne 0 ]
   do
      r=`expr $num % 10`
      sum=`expr $(($sum * 10)) + $r`
      num=`expr $num / 10`
   done
   if [ $1 -eq $sum ]
   then
      return 0
   else
      return 1
   fi
}
arr=()
index=0
for i in `seq 10 100`
do
   if isPalindrome $i
   then
      arr[((index++))]=$i
   fi
done
echo ${arr[@]}
```

```
+ arr=()
+ index=0
++ seq 10 20
+ for i in `seq 10 20`
+ isPalindrome 10
+ num=10
+ sum=0
+ '[' 10 -ne 0 ']'
++ expr 10 % 10
+ r=0
++ expr 0 + 0
+ sum=0
++ expr 10 / 10
+ num=1
+ '[' 1 -ne 0 ']'
++ expr 1 % 10
+ r=1
++ expr 0 + 1
+ sum=1
++ expr 1 / 10
+ num=0
+ '[' 0 -ne 0 ']'
+ '[' 10 -eq 1 ']'
+ return 1
+ for i in `seq 10 20`
+ isPalindrome 11
+ num=11
+ sum=0
+ '[' 11 -ne 0 ']'
++ expr 11 % 10
+ r=1
+ sum=11
++ expr 1 / 10
+ num=0
+ '[' 0 -ne 0 ']'
+ '[' 11 -eq 11 ']'
+ return 0
+ arr[((index++))]=11
+ for i in `seq 10 20`
+ isPalindrome 12
+ num=12
+ sum=0
+ '[' 12 -ne 0 ']'
++ expr 12 % 10
+ r=2
++ expr 0 + 2
+ sum=2
++ expr 12 / 10
+ num=1
+ '[' 1 -ne 0 ']'
++ expr 1 % 10
+ r=1
++ expr 20 + 1
+ sum=21
++ expr 1 / 10
+ num=0
+ '[' 0 -ne 0 ']'
+ '[' 12 -eq 21 ']'
+ return 1
+ for i in `seq 10 20`
+ isPalindrome 13
+ num=13
+ sum=0
+ '[' 13 -ne 0 ']'
++ expr 13 % 10
+ r=3
++ expr 0 + 3
+ sum=3
++ expr 13 / 10
+ num=1
+ '[' 1 -ne 0 ']'
++ expr 1 % 10
+ r=1
++ expr 30 + 1
+ sum=31
++ expr 1 / 10
+ num=0
+ '[' 0 -ne 0 ']'
++ expr 0 + 1
+ sum=1
++ expr 11 / 10
+ num=1
+ '[' 1 -ne 0 ']'
++ expr 1 % 10
+ r=1
++ expr 10 + 1
+ '[' 13 -eq 31 ']'
+ return 1
+ for i in `seq 10 20`
+ isPalindrome 14
+ num=14
+ sum=0
+ '[' 14 -ne 0 ']'
++ expr 14 % 10
+ r=4
++ expr 0 + 4
+ sum=4
++ expr 14 / 10
+ num=1
+ '[' 1 -ne 0 ']'
++ expr 1 % 10
+ r=1
++ expr 40 + 1
+ sum=41
++ expr 1 / 10
+ num=0
+ '[' 0 -ne 0 ']'
+ '[' 14 -eq 41 ']'
+ return 1
+ for i in `seq 10 20`
+ isPalindrome 15
+ num=15
+ sum=0
+ '[' 15 -ne 0 ']'
++ expr 15 % 10
+ r=5
++ expr 0 + 5
+ sum=5
++ expr 15 / 10
+ num=1
+ '[' 1 -ne 0 ']'
++ expr 1 % 10
+ r=1
++ expr 50 + 1
+ sum=51
++ expr 1 / 10
+ num=0
+ '[' 0 -ne 0 ']'
+ '[' 15 -eq 51 ']'
+ return 1
+ for i in `seq 10 20`
+ isPalindrome 16
+ num=16
+ sum=0
+ '[' 16 -ne 0 ']'
++ expr 16 % 10
+ r=6
++ expr 0 + 6
+ sum=6
++ expr 16 / 10
+ num=1
+ '[' 1 -ne 0 ']'
++ expr 1 % 10
+ r=1
++ expr 60 + 1
+ sum=61
++ expr 1 / 10
+ num=0
+ '[' 0 -ne 0 ']'
+ '[' 16 -eq 61 ']'
+ return 1
+ for i in `seq 10 20`
+ isPalindrome 17
+ num=17
+ sum=0
+ '[' 17 -ne 0 ']'
++ expr 17 % 10
+ r=7
++ expr 0 + 7
+ sum=7
++ expr 17 / 10
+ num=1
+ '[' 1 -ne 0 ']'
++ expr 1 % 10
+ r=1
++ expr 70 + 1
+ sum=71
++ expr 1 / 10
+ num=0
+ '[' 0 -ne 0 ']'
+ '[' 17 -eq 71 ']'
+ return 1
+ for i in `seq 10 20`
+ isPalindrome 18
+ num=18
+ sum=0
+ '[' 18 -ne 0 ']'
++ expr 18 % 10
+ r=8
++ expr 0 + 8
+ sum=8
++ expr 18 / 10
+ num=1
+ '[' 1 -ne 0 ']'
++ expr 1 % 10
+ r=1
++ expr 80 + 1
+ sum=81
++ expr 1 / 10
+ num=0
+ '[' 0 -ne 0 ']'
+ '[' 18 -eq 81 ']'
+ return 1
+ for i in `seq 10 20`
+ isPalindrome 19
+ num=19
+ sum=0
+ '[' 19 -ne 0 ']'
++ expr 19 % 10
+ r=9
++ expr 0 + 9
+ sum=9
++ expr 19 / 10
+ num=1
+ '[' 1 -ne 0 ']'
++ expr 1 % 10
+ r=1
++ expr 90 + 1
+ sum=91
++ expr 1 / 10
+ num=0
+ '[' 0 -ne 0 ']'
+ '[' 19 -eq 91 ']'
+ return 1
+ for i in `seq 10 20`
+ isPalindrome 20
+ num=20
+ sum=0
+ '[' 20 -ne 0 ']'
++ expr 20 % 10
+ r=0
++ expr 0 + 0
+ sum=0
++ expr 20 / 10
+ num=2
+ '[' 2 -ne 0 ']'
++ expr 2 % 10
+ r=2
++ expr 0 + 2
+ sum=2
++ expr 2 / 10
+ num=0
+ '[' 0 -ne 0 ']'
+ '[' 20 -eq 2 ']'
+ return 1
+ echo 11
11
```