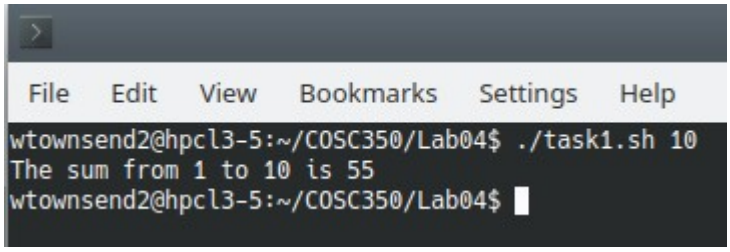
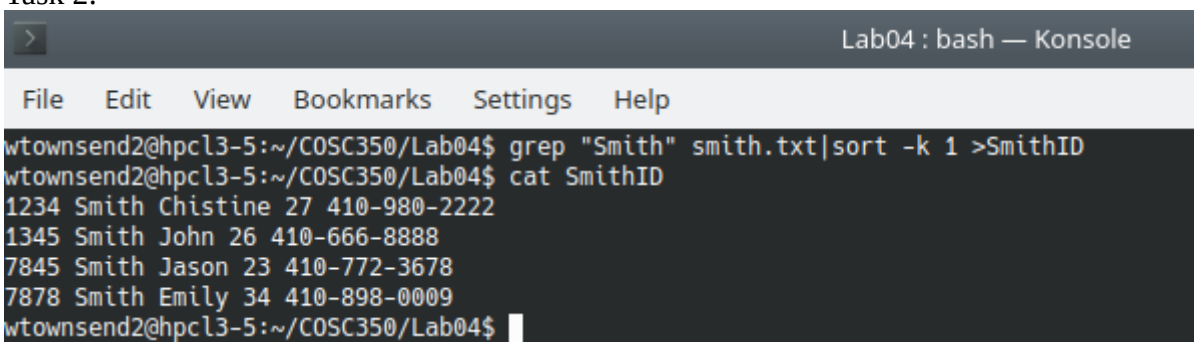
Task 1:



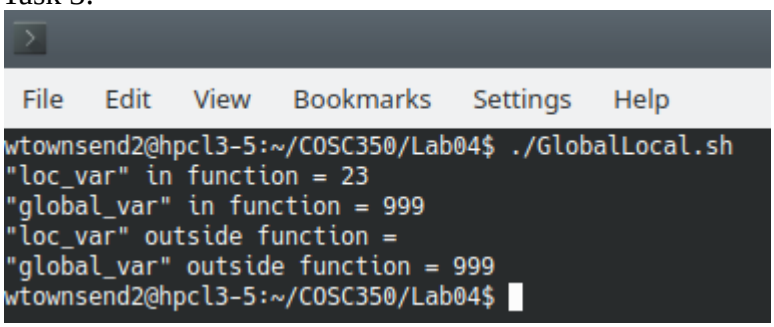

```
#!/bin/bash
sum=0
for i in $(seq 1 $1);do
    sum=`expr $sum + $i`
done

echo "The sum from 1 to $1 is $sum"
exit 0
```

Task 2:

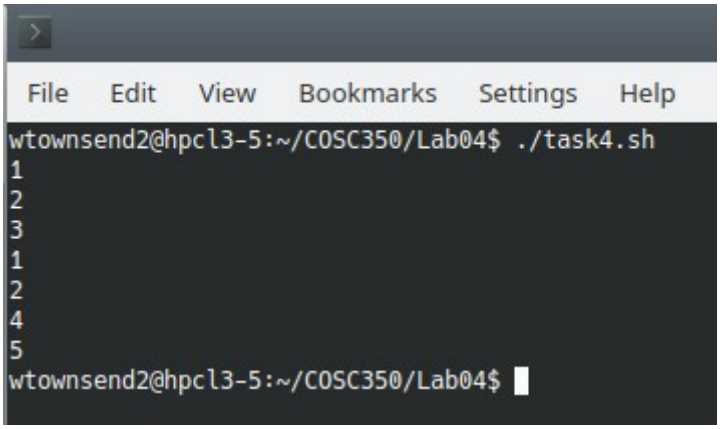

Lab04 : bash — Konsole
File Edit View Bookmarks Settings Help
wtownsend2@hpcl3-5:~/COSC350/Lab04$ grep "Smith" smith.txt|sort -k 1 >SmithID
wtownsend2@hpcl3-5:~/COSC350/Lab04$ cat SmithID
1234 Smith Chistine 27 410-980-2222
1345 Smith John 26 410-666-8888
7845 Smith Jason 23 410-772-3678
7878 Smith Emily 34 410-898-0009
wtownsend2@hpcl3-5:~/COSC350/Lab04$

Task 3:




```
#!/bin/bash
#LECTURE SLIDES
# GlobalLocal.sh
# Global and local variables inside a function.
LocalGlobal ()
{
local loc_var=23 # Declared as local variable.
echo "\"loc_var\" in function = $loc_var"
global_var=999 #global variable
echo "\"global_var\" in function = $global_var"
}
LocalGlobal
echo "\"loc_var\" outside function = $loc_var"
echo "\"global_var\" outside function = $global_var"
exit 0
```

Task 4:



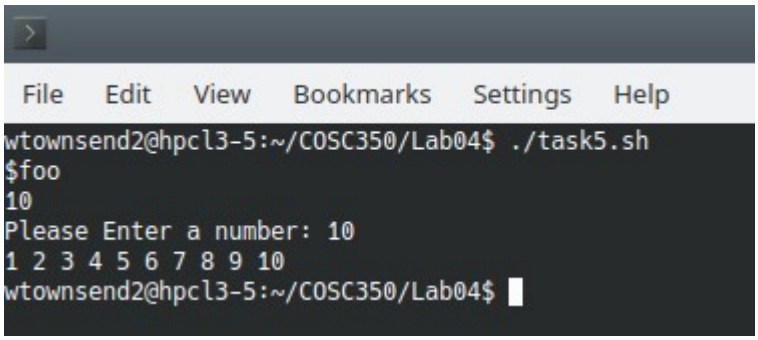

```
#!/bin/bash
#LECTURE SLIDES
for myloop in 1 2 3 4 5
do
   echo "$myloop"
   if [ "$myloop" -eq 3 ]
   then
      break
   fi
done

for myloop in 1 2 3 4 5
do
   if [ "$myloop" -eq 3 ]
   then
   continue # Skip rest of loop iteration.
   fi
   echo "$myloop"
done

exit 0
```

Break completely exits the loop and moving on to the rest of the program, while continue only exits the current cycle ignoring everything after the keyword but continues on to the next cycle of the loop.

Task 5:



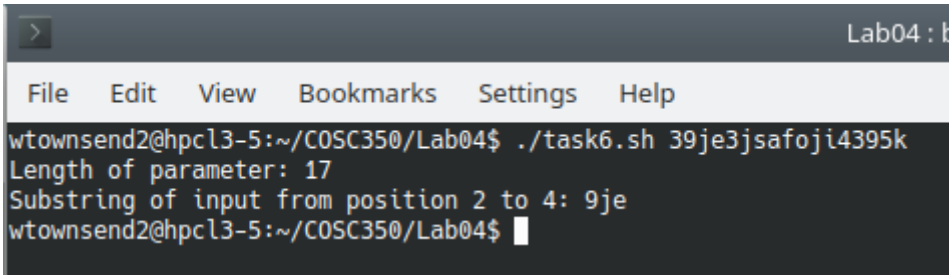

```
#LECTURE SLIDES
foo=10
x=foo
y=\$$x #it is same as y='$'$x
echo $y
foo=10
x=foo
eval y=\$$x
echo $y

#Experimental Code
echo -n "Please Enter a number: "
read num;
ref=num
eval y=\$$ref
for i in $(seq 1 $y);do
    echo -n "$i "
done
echo

exit 0
```

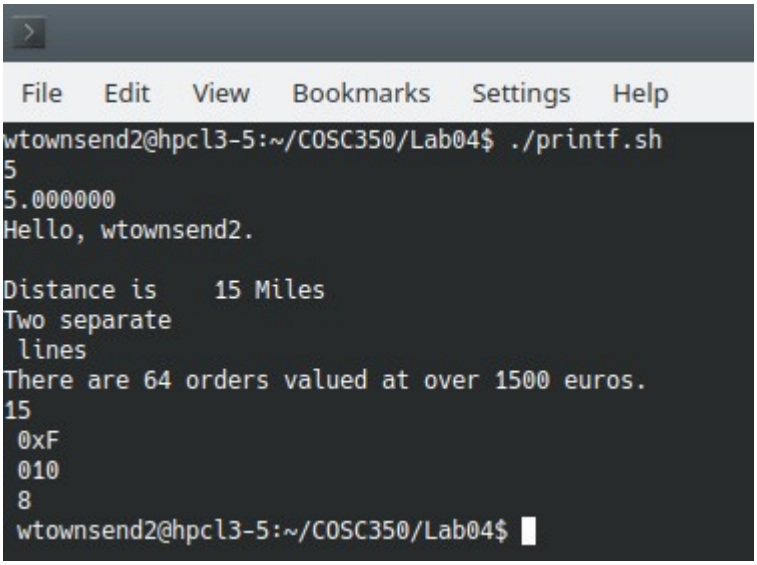
Eval lets the program indirectly reference a variable and evaluate it as such

Task 6:



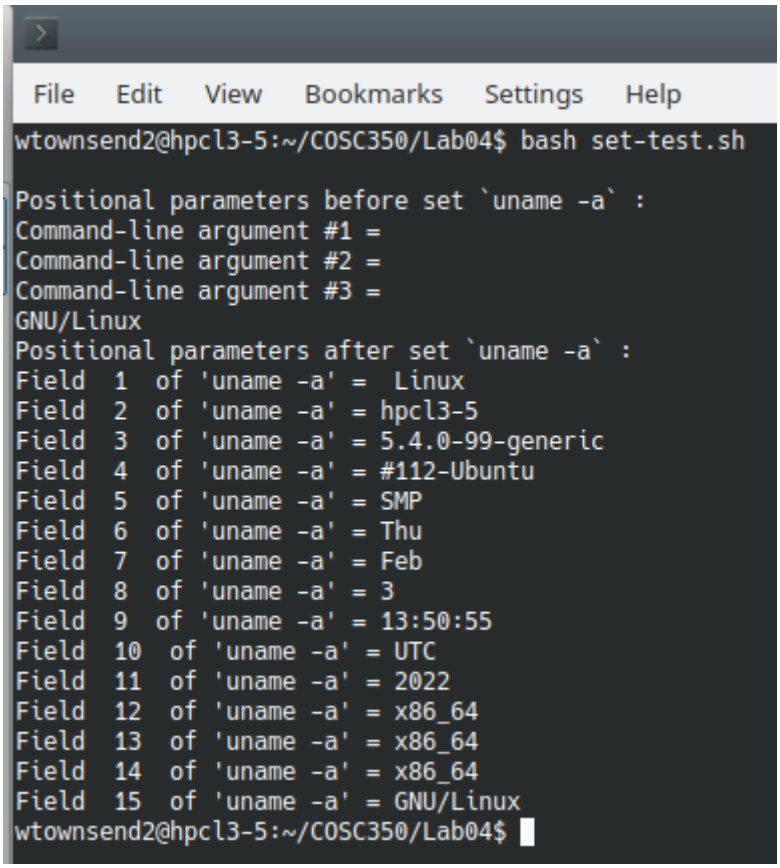

```
#!/bin/bash
if [ "$#" -ne 1 ];then
    echo "Please include one parameter!"
    exit 1
fi
len=`expr length $1`
sub=`expr substr $1 2 3`
echo "Length of parameter: $len"
echo "Substring of input from position 2 to 4: $sub"
exit 0
```

Task 7:




```
#!/bin/sh
#LECTURE SLIDES
# printf.sh
#Print the decimal number 5 followed by a newline (\n)
printf "%d\n" 5
#Print as float (default 6 decimal places)
printf "%f\n" 5
#Print text followed by variable $USER
printf "Hello, $USER.\n\n"
#Display variables distance=15 with 5 spaces aligned to
distance=15
printf "Distance is %5d Miles\n" $distance
#Use \n anywhere to start a new line:
printf "Two separate\n lines \n"
#Print decimal numbers interspersed with text
printf "There are %d orders valued at over %d euros.\n" 64 1500
#Convert a hex number to decimal
printf "%d\n " 0xF
#Convert a decimal number to Hex
printf "0x%X\n " 15
#Convert a decimal number to Octal
printf "0%o\n " 8
#Convert an Octal number to decimal
printf "%d\n " 010
exit 0
```

Task 8:




```
#!/bin/sh
#LECTURE SLIDES
# script "set-test1.sh"
# Invoke this script with three command line parameters,
# for example, "./set-test one two three".
echo
echo "Positional parameters before set \`uname -a\` :"
echo "Command-line argument #1 = $1"
echo "Command-line argument #2 = $2"
echo "Command-line argument #3 = $3"
set `uname -a` # Sets the positional parameters to the output
# of the command `uname -a`
# The output of `uname -a`
# Linux sophie-wang 5.4.0-100-generic #113-Ubuntu SMP Thu Feb 3 18:43:29 UTC 2022 x86_64
x86_64 x86_64 GNU/Linux
echo $_ # last positional parameter
# Flags set in script.
echo "Positional parameters after set \`uname -a\` :"
# $1, $2, $3, etc. reinitialized to result of `uname -a`
i=1
for ARG in " $@ "; do
    echo "Field " $i " of 'uname -a' = $ARG"
    let i++
done
exit 0
```