DTS Unix-Shell Scripting Capstone Assignment

Name: Atharva Athanikar 72287317E

Batch: DTS Batch-3

1) Write a shell script that folds long lines into 40 columns. Thus, any line that exceeds 40 characters must be broken after 40th, a "\" is to be appended as the indication of folding and the processing is to be continued with the residue. The input is to be supplied through a text file created by the user.

Code:

```
#Program 1
input_file=$1
while IFS= read -r line; do
  while [ ${#line} -gt 40 ]; do
    echo "${line:0:40}\\"
    line="${line:40}"
  done
  echo "$line"
done < "$input_file"
```

OUTPUT:

MINGW64:/c/Users/athar/script25

```
#Program 1

input_file=$1
if [ -z "$input_file" ]; then
  echo "Please provide an input file."
  exit 1
fi

if [ ! -f "$input_file" ]; then
  echo "File '$input_file' not found!"
  exit 1
fi
while IFS= read -r line; do
  while [ ${#line} -gt 40 ]; do
    echo "${line:0:40}\\"
    line="${line:40}"
  done
  echo "$line"
done < "$input_file"
```

MINGW64:/c/Users/athar/script25

```
athar@ASUSVivobook MINGW64 ~/script25 (master)
$ vi folds.sh

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ cat input_file.txt
Write a shell script that folds long lines into 40 columns. Thus any line that
exceeds 40 characters must be broken after 40th, a “\” is to be appended as the
indication of folding and the processing is to be continued with the residue. The
input is to be supplied through a text file created by the user.

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ sh folds.sh input_file.txt
Write a shell script that folds long lin\
es into 40 columns. Thus any line that
exceeds 40 characters must be broken aft\
er 40th, a “\” is to be appended as the
indication of folding and the processing\
 is to be continued with the residue. Th\
e
input is to be supplied through a text f\
ile created by the user.

athar@ASUSVivobook MINGW64 ~/script25 (master)
$
```

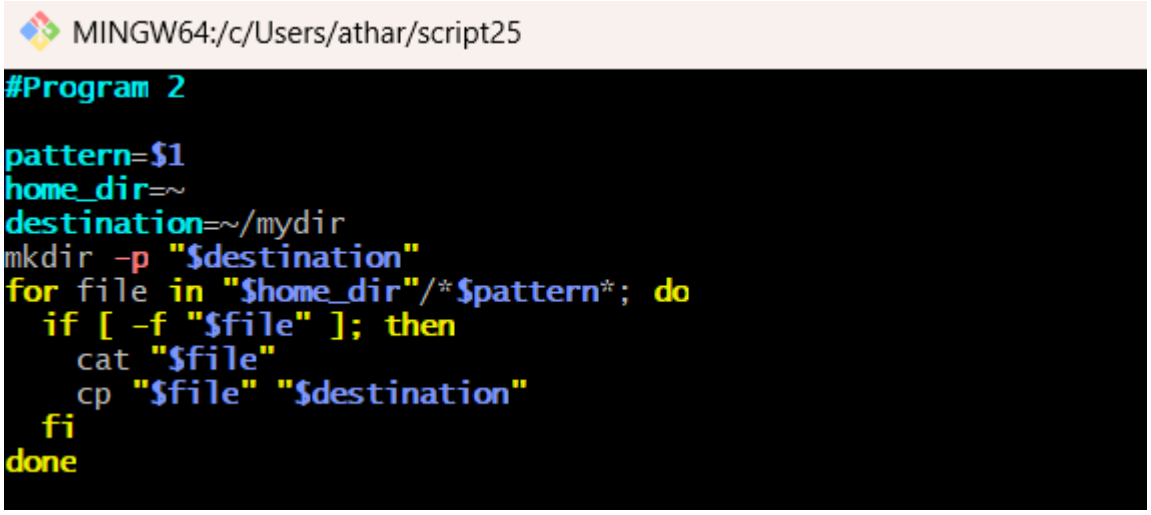
2) Write a shell script to find a file/s that matches a pattern given as command

line argument in the home directory, display the contents of the file and copy the file into the directory ~/mydir.

Code:

```
#Program 2
pattern=$1
home_dir=~
destination=~/mydir
mkdir -p "$destination"
for file in "$home_dir"/*$pattern*; do
  if [ -f "$file" ]; then
    cat "$file"
    cp "$file" "$destination"
  fi
done
```

OUTPUT:

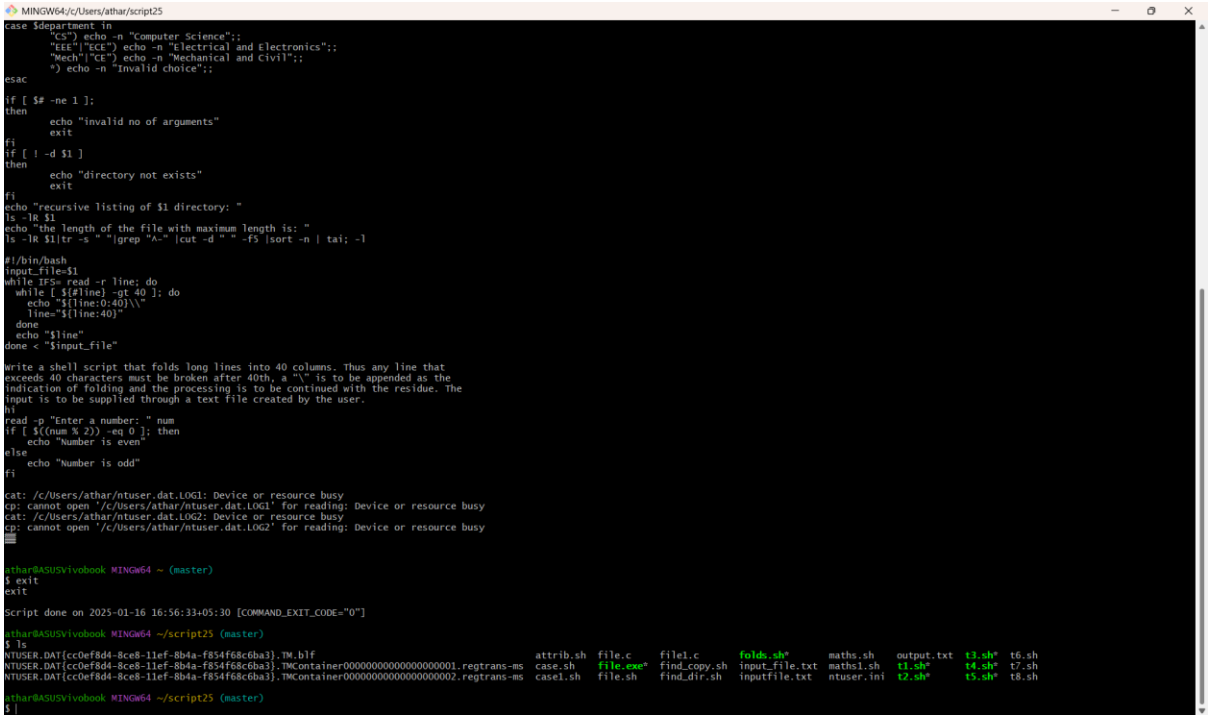


MINGW64:/c/Users/athar/script25

```
athar@ASUSVivobook MINGW64 ~/script25 (master)
$ vi find_copy.sh

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ sh find_copy.sh
cat: /c/Users/athar/NTUSER.DAT: Device or resource busy
cp: cannot open '/c/Users/athar/NTUSER.DAT' for reading: Device or resource busy
x[illegible]p[illegible]z[illegible]PO--[illegible]x
[illegible]\Device\HarddiskVolume3\Users\athar\NTUSER.DAT{cc0ef8d4-8ce8-11ef-8b4a-f854f68c6ba
3}.TM.blf[illegible]0$[illegible]8LF$\NTUSER.DAT{cc0ef8d4-8ce8-11ef-8b4a-f854f68c6ba3}.TMContainer00000000000000000
1.regtrans-ms[illegible]0$8[illegible]8LF$\NTUSER.DAT{cc0ef8d4-8ce8-11ef-8b4a-f854f68c6ba3}.TMContainer00000000000000
0000002.regtrans-ms00--[illegible]y
[illegible]\Device\HarddiskVolume3\Users\athar\NTUSER.DAT{cc0ef8d4-8ce8-11ef-8b4a-f854f68c6ba3}.T
M.blf[illegible]0$8[illegible]8LF$\NTUSER.DAT{cc0ef8d4-8ce8-11ef-8b4a-f854f68c6ba3}.TMContainer00000000000000000001.re
gtrans-ms[illegible]0$8[illegible]8LF$\NTUSER.DAT{cc0ef8d4-8ce8-11ef-8b4a-f854f68c6ba3}.TMContainer00000000000000000
002.regtrans-ms00[illegible]p[illegible]pp[illegible]H(&[illegible]&[illegible]T[illegible]kdecho "enter a filename: "
read fn
if [ -f $fn ]
then
        ls -l $fn
else
        echo "file not found"
fi

echo "MENU\n 1 List of files\n 2.User Processes\n 3. Today's Date\n 4.Users of system\n 5.Quit\n Enter
your option"
read choice
case "$choice" in
        1) ls -l ;;
        2) ps -f ;;
        3) date ;;
        4) who ;;
        5) exit ;;
        6) echo "Invalid option" ;;
esac

echo "enter your department: "
read department
echo -n "Your department is: "
case $department in
        "CS") echo -n "Computer Science";;
        "EEE"|"ECE") echo -n "Electrical and Electronics";;
        "Mech"|"CE") echo -n "Mechanical and Civil";;
        *) echo -n "Invalid choice";;
esac

if [ $# -ne 1 ];
then
        echo "invalid no of arguments"
        exit
fi
if [ ! -d $1 ]
then
        echo "directory not exists"
        exit
fi
echo "recursive listing of $1 directory: "
ls -lR $1
echo "the length of the file with maximum length is: "
ls -lR $1|tr -s " "|grep "^-" |cut -d " " -f5 |sort -n | tai; -l

#!/bin/bash
input_file=$1
while IFS= read -r line; do
```

MINGW64:/c/Users/athar/script25

```
        6) echo "Invalid option" ;;
esac

echo "enter your department: "
read department
echo -n "Your department is: "
case $department in
        "CS") echo -n "Computer Science";;
        "EEE"|"ECE") echo -n "Electrical and Electronics";;
        "Mech"|"CE") echo -n "Mechanical and Civil";;
        *) echo -n "Invalid choice";;
esac

if [ $# -ne 1 ];
then
        echo "invalid no of arguments"
        exit
fi
if [ ! -d $1 ]
then
        echo "directory not exists"
        exit
fi
echo "recursive listing of $1 directory: "
ls -lR $1
echo "the length of the file with maximum length is: "
ls -lR $1|tr -s " "|grep "^-" |cut -d " " -f5 |sort -n | tai; -l

#!/bin/bash
input_file=$1
while IFS= read -r line; do
  while [ ${#line} -gt 40 ]; do
    echo "${line:0:40}\\"
    line="${line:40}"
  done
  echo "$line"
done < "$input_file"

Write a shell script that folds long lines into 40 columns. Thus any line that
exceeds 40 characters must be broken after 40th, a "\" is to be appended as the
indication of folding and the processing is to be continued with the residue. The
input is to be supplied through a text file created by the user.
hi
read -p "Enter a number: " num
if [ $((num % 2)) -eq 0 ]; then
    echo "Number is even"
else
    echo "Number is odd"
fi

cat: /c/Users/athar/ntuser.dat.LOG1: Device or resource busy
cp: cannot open '/c/Users/athar/ntuser.dat.LOG1' for reading: Device or resource busy
cat: /c/Users/athar/ntuser.dat.LOG2: Device or resource busy
cp: cannot open '/c/Users/athar/ntuser.dat.LOG2' for reading: Device or resource busy
[illegible]


athar@ASUSVivobook MINGW64 ~ (master)
$ exit
exit

Script done on 2025-01-16 16:56:33+05:30 [COMMAND_EXIT_CODE="0"]

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ |
```



```
MINGW64:/c/Users/athar/script25
case $department in
        "CS") echo -n "Computer Science";;
        "EEE"|"ECE") echo -n "Electrical and Electronics";;
        "Mech"|"CE") echo -n "Mechanical and Civil";;
        *) echo -n "Invalid choice";;
esac

if [ $# -ne 1 ];
then
        echo "invalid no of arguments"
        exit
fi
if [ ! -d $1 ]
then
        echo "directory not exists"
        exit
fi
echo "recursive listing of $1 directory: "
ls -lR $1
echo "the length of the file with maximum length is: "
ls -lR $1|tr -s " "|grep "^-" |cut -d " " -f5 |sort -n | tai; -l

#!/bin/bash
input_file=$1
while IFS= read -r line; do
  while [ ${#line} -gt 40 ]; do
    echo "${line:0:40}\\"
    line="${line:40}"
  done
  echo "$line"
done < "$input_file"

Write a shell script that folds long lines into 40 columns. Thus any line that
exceeds 40 characters must be broken after 40th, a "\" is to be appended as the
indication of folding and the processing is to be continued with the residue. The
input is to be supplied through a text file created by the user.
hi
read -p "Enter a number: " num
if [ $((num % 2)) -eq 0 ]; then
    echo "Number is even"
else
    echo "Number is odd"
fi

cat: /c/Users/athar/ntuser.dat.LOG1: Device or resource busy
cp: cannot open '/c/Users/athar/ntuser.dat.LOG1' for reading: Device or resource busy
cat: /c/Users/athar/ntuser.dat.LOG2: Device or resource busy
cp: cannot open '/c/Users/athar/ntuser.dat.LOG2' for reading: Device or resource busy

athar@ASUSVivobook MINGW64 ~ (master)
$ exit
exit

Script done on 2025-01-16 16:56:33+05:30 [COMMAND_EXIT_CODE="0"]

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ ls
NTUSER.DAT{cc0ef8d4-8ce8-11ef-8b4a-f854f68c6ba3}.TM.blf                    attrib.sh  file.c     file1.c       folds.sh*     maths.sh    output.txt  t3.sh*  t6.sh
NTUSER.DAT{cc0ef8d4-8ce8-11ef-8b4a-f854f68c6ba3}.TMContainer00000000000000000001.regtrans-ms  case.sh    file.exe*  find_copy.sh  input_file.txt  maths1.sh   t1.sh*      t4.sh*  t7.sh
NTUSER.DAT{cc0ef8d4-8ce8-11ef-8b4a-f854f68c6ba3}.TMContainer00000000000000000002.regtrans-ms  case1.sh   file.sh    find_dir.sh   inputfile.txt   ntuser.ini  t2.sh*      t5.sh*  t8.sh

athar@ASUSVivobook MINGW64 ~/script25 (master)
$
```

3) Write a shell script to compute GCD & LCM of two numbers.

<u>Code:</u>

```
#Program 3
gcd() {
  while [ $2 -ne 0 ]; do
    t=$2
    b=$(( $1 % $2 ))
    set -- $t $b
  done
  echo $1
}
lcm() {
  echo $(( $1 * $2 / $(gcd $1 $2) ))
}
read -p "Enter the first number: " num1
read -p "Enter the second number: " num2
```

```
if [[ ! $num1 =~ ^[0-9]+$ || ! $num2 =~ ^[0-9]+$ ]]; then
  echo "Invalid input. Please enter positive integers."
  exit 1
fi
g=$(gcd $num1 $num2)
l=$(lcm $num1 $num2)
echo "GCD: $g, LCM: $l"
```

OUTPUT:

MINGW64:/c/Users/athar/script25

```
#Program 3

gcd() {
  while [ $2 -ne 0 ]; do
    t=$2
    b=$(( $1 % $2 ))
    set -- $t $b
  done
  echo $1
}
lcm() {
  echo $(( $1 * $2 / $(gcd $1 $2) ))
}
read -p "Enter the first number: " num1
read -p "Enter the second number: " num2
if [[ ! $num1 =~ ^[0-9]+$ || ! $num2 =~ ^[0-9]+$ ]]; then
  echo "Invalid input. Please enter positive integers."
  exit 1
fi
g=$(gcd $num1 $num2)
l=$(lcm $num1 $num2)
echo "GCD: $g, LCM: $l"
```

MINGW64:/c/Users/athar/script25

```
athar@ASUSVivobook MINGW64 ~/script25 (master)
$ vi gcd_lcm.sh

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ sh gcd_lcm.sh
Enter the first number: 2
Enter the second number: 54
GCD: 2, LCM: 54

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ sh gcd_lcm.sh
Enter the first number: 12
Enter the second number: 18
GCD: 6, LCM: 36

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ sh gcd_lcm.sh
Enter the first number: 10
Enter the second number: 25
GCD: 5, LCM: 50

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ |
```

4) Write a shell script that accepts a pathname and creates all the components in that pathname as directories. for example, if the script it named mpc,then the command mpc a/b/c/d should create directories a,a/b,a/b/c,a/b/c/d.

Code:

```
#Program 4
if [ $# -eq 0 ]; then
  echo "Usage: $0 pathname"
  exit 1
fi
mkdir -p "$1"
```

OUTPUT:

MINGW64:/c/Users/athar/script25

```
#Program 4

if [ $# -eq 0 ]; then
  echo "Usage: $0 pathname"
  exit 1
fi
mkdir -p "$1"
```

MINGW64:/c/Users/athar/script25

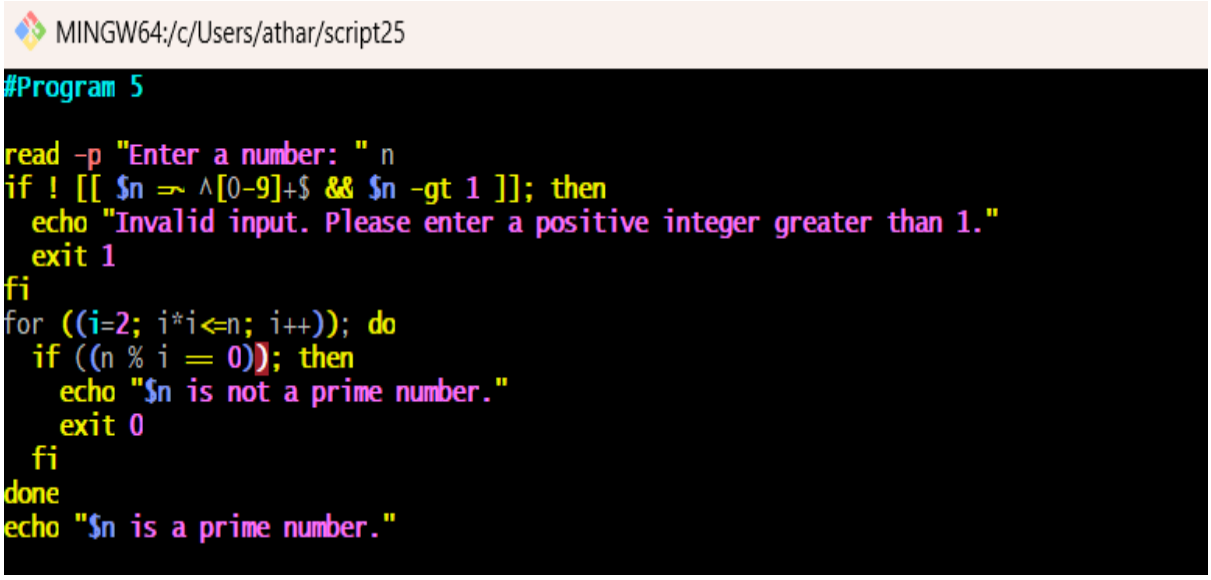
```
athar@ASUSVivobook MINGW64 ~/script25 (master)
$ vi pathname.sh

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ sh pathname.sh
Usage: pathname.sh pathname

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ ./pathname.sh a/b/c/d

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ ls -R a
a:
b/

a/b:
c/

a/b/c:
d/

a/b/c/d:

athar@ASUSVivobook MINGW64 ~/script25 (master)
$
```

5) Write a shell script to find whether a given number is prime.

Code:

```
#Program 5
read -p "Enter a number: " n
if ! [[ $n =~ ^[0-9]+$ && $n -gt 1 ]]; then
  echo "Invalid input. Please enter a positive integer greater than 1."
  exit 1
fi
for ((i=2; i*i<=n; i++)); do
  if ((n % i == 0)); then
    echo "$n is not a prime number."
    exit 0
  fi
done
echo "$n is a prime number."
```

OUTPUT:



```
MINGW64:/c/Users/athar/script25

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ vi prime.sh

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ sh prime.sh
Enter a number: 5
5 is a prime number.

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ sh prime.sh
Enter a number: 51
51 is not a prime number.

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ sh prime.sh
Enter a number: 1
Invalid input. Please enter a positive integer greater than 1.

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ |
```

6) Write a shell script that get executed display the message either “good morning” or “good afternoon” or “good evening” depending upon time at which the user logs-in.

<u>Code:</u>

```
#Program 6
hour=$(date +%H)
if [ $hour -lt 12 ]; then
  echo "Good morning"
elif [ $hour -lt 18 ]; then
  echo "Good afternoon"
else
  echo "Good evening"
fi
```

OUTPUT:

```
MINGW64:/c/Users/athar/script25
#Program 6

hour=$(date +%H)
if [ $hour -lt 12 ]; then
  echo "Good morning"
elif [ $hour -lt 18 ]; then
  echo "Good afternoon"
else
  echo "Good evening"
fi
```

```
MINGW64:/c/Users/athar/script25

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ vi greeting.sh

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ sh greeting.sh
Good afternoon

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ |
```

7) Write a shell script that accepts two file names as arguments, check if the permissions for these files are identical and if the permissions are identical, output common permissions and otherwise output each file names followed by it's permissions.

Code:

```
# Program 7
if [ $# -ne 2 ]; then
  echo "Usage: $0 file1 file2"
  exit 1
fi
perm1=$(stat -c %A "$1")
perm2=$(stat -c %A "$2")
```

```
if [ "$perm1" = "$perm2" ]; then
  echo "Identical permissions: $perm1"
else
  echo "$1: $perm1"
  echo "$2: $perm2"
fi
```

OUTPUT:

MINGW64:/c/Users/athar/script25

```
#!/bin/bash
if [ $# -ne 2 ]; then
  echo "Usage: $0 file1 file2"
  exit 1
fi
perm1=$(stat -c %A "$1")
perm2=$(stat -c %A "$2")
if [ "$perm1" = "$perm2" ]; then
  echo "Identical permissions: $perm1"
else
  echo "$1: $perm1"
  echo "$2: $perm2"
fi
```

MINGW64:/c/Users/athar/script25

```
athar@ASUSVivobook MINGW64 ~/script25 (master)
$ vi compare.sh

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ sh compare.sh t1.sh t6.sh
t1.sh: -rwxr-xr-x
t6.sh: -rw-r--r--

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ sh compare.sh t5.sh t6.sh
t5.sh: -rwxr-xr-x
t6.sh: -rw-r--r--

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ sh compare.sh prime.sh compare.sh
prime.sh: -rw-r--r--
compare.sh: -rwxr-xr-x

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ ./compare.sh maths.sh file1.c
Identical permissions: -rw-r--r--

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ ./compare.sh maths.sh input_file.txt
Identical permissions: -rw-r--r--

athar@ASUSVivobook MINGW64 ~/script25 (master)
$ |
```