Experiment - 9
SHELL PROGRAMMING

Name -Rudram Sharma
Batch -9
Sap id - 590021706

Aim:

1. Write a script that renames all files in a directory by adding a prefix or suffix to the filenames.

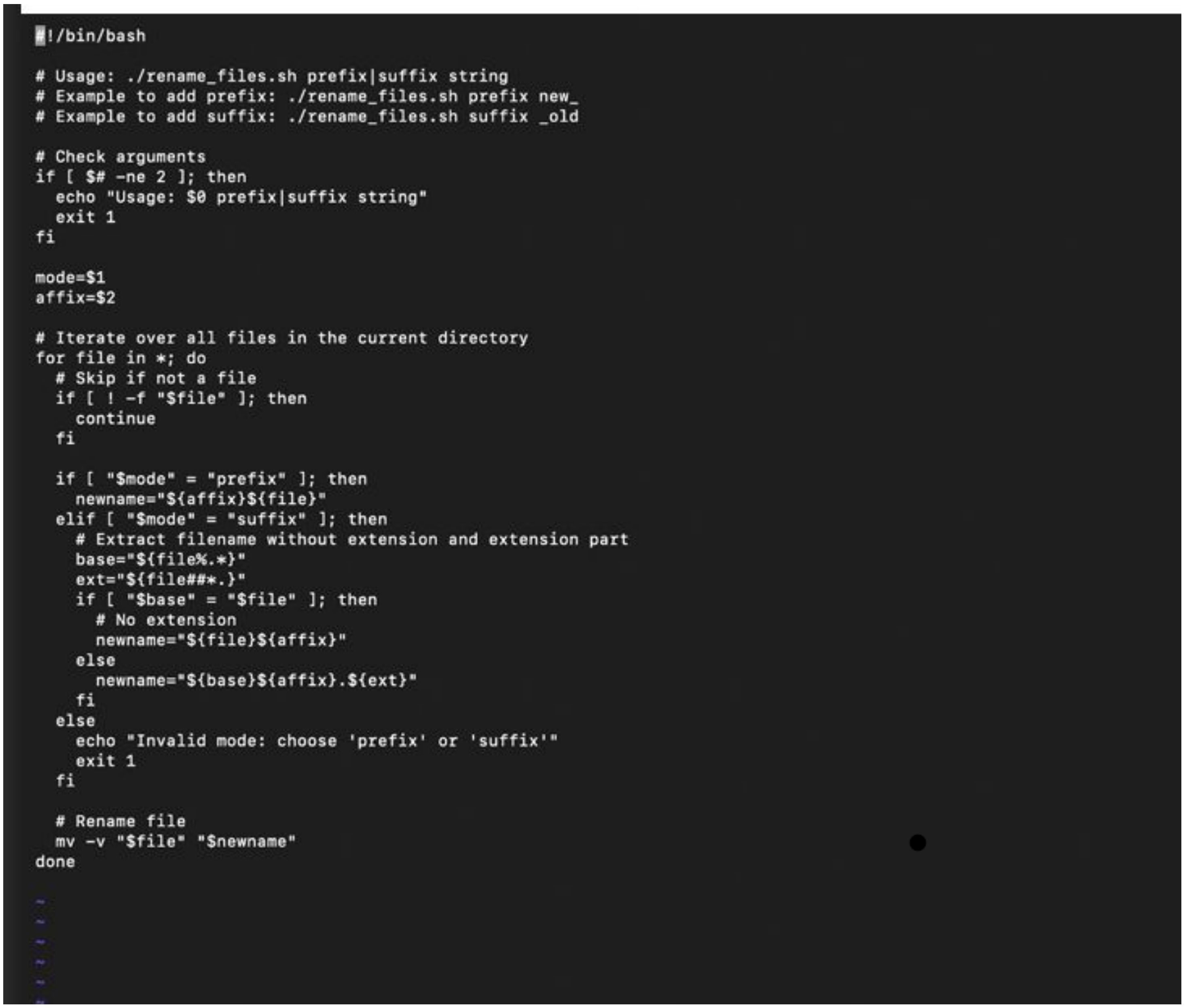
2. Create a script that searches for files in a specified directory and its subdirectories, based on certain criteria like file extension or file size.

3. Write a script that generates the Fibonacci series up to a given number, using loops or recursive functions.

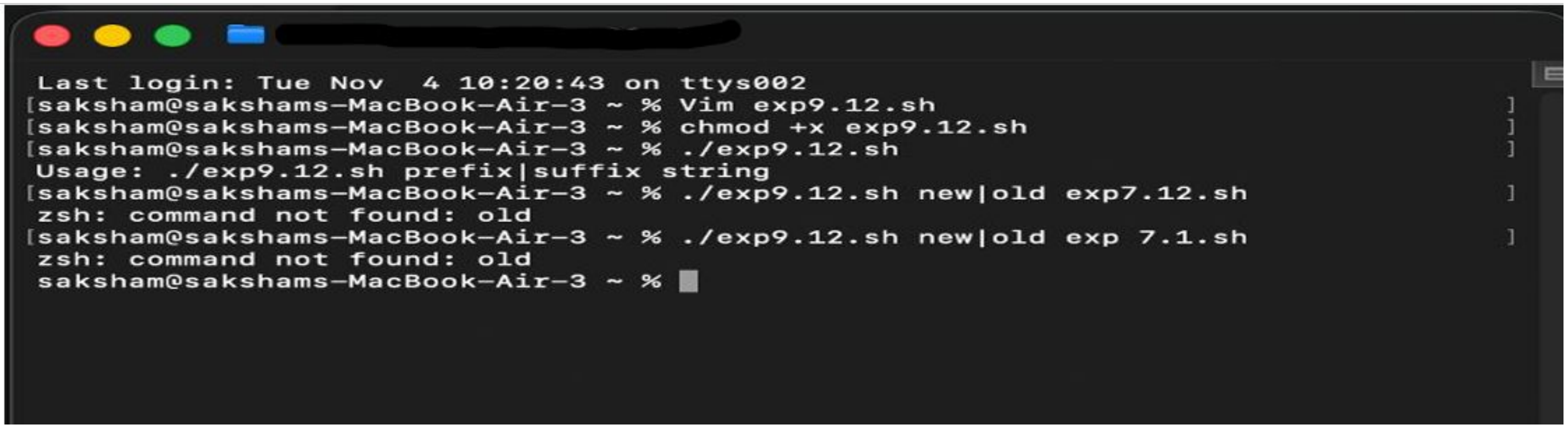
Theory

- Shell scripting on Linux is used to automate routine and multi-step file operations by executing a curated sequence of commands, turning repetitive tasks into reliable workflows.
- Execution privileges may need to be granted with chmod so that scripts run securely and as intended under the current user or environment.
- File discovery typically uses find, grep, or ls with filters to match by extension, size, or name patterns, enabling targeted searches across directories
- Pipelines (|) and redirection (>, >>) enable chaining tools together and capturing results to files for later processing or reporting.
- Recursive traversal across subdirectories ensures comprehensive scanning of a directory tree so no relevant files are missed.
- The Fibonacci sequence is a classic exercise to strengthen control-flow reasoning in shell, contrasting iterative loops with function-based recursion.
- Simple recursion can be modeled with bash functions, allowing comparison between iterative and recursive strategies for similar logic.

1. Script that renames all files in a directory by adding a prefix or suffix to the filenames.



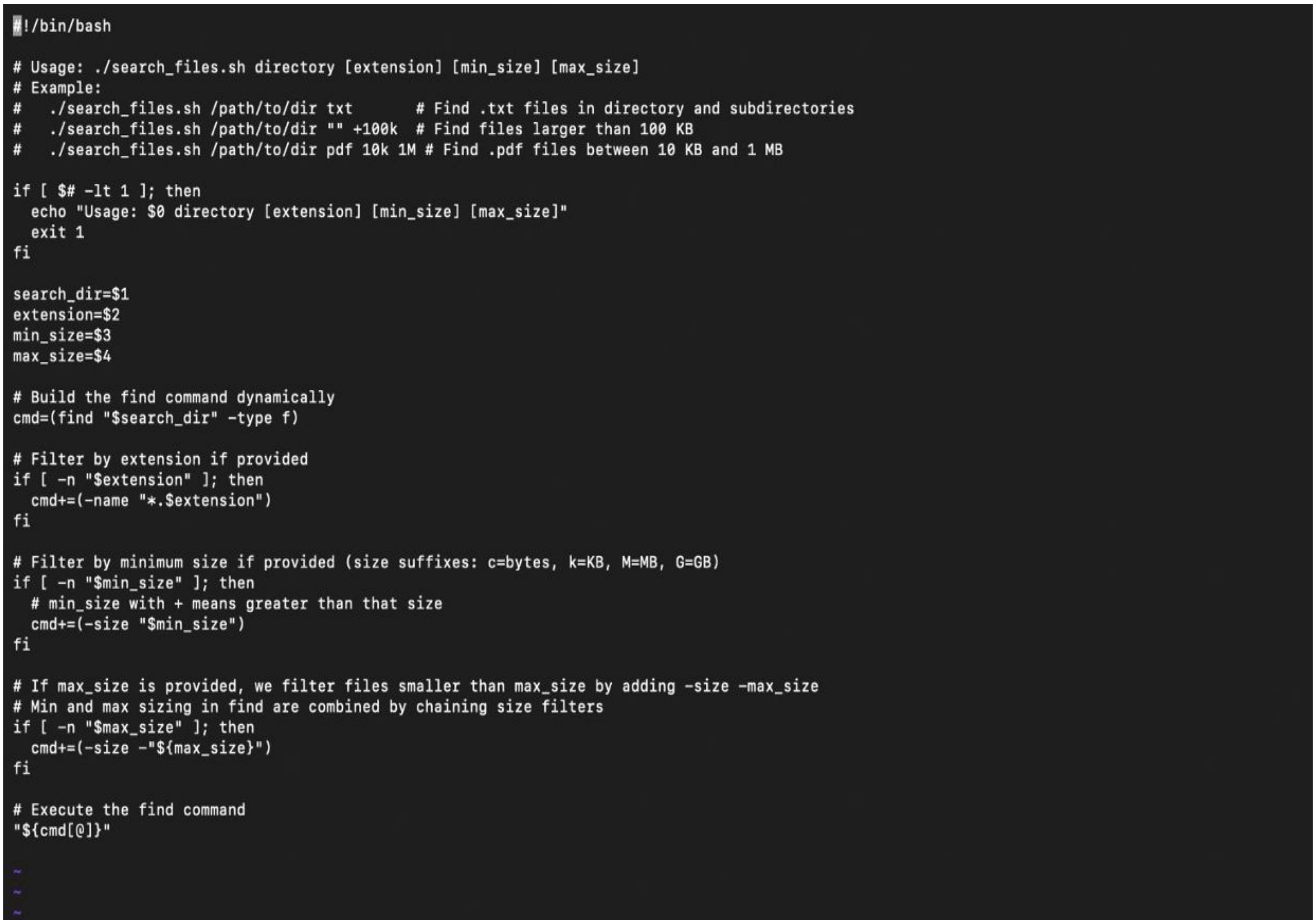
```
#!/bin/bash

# Usage: ./rename_files.sh prefix|suffix string
# Example to add prefix: ./rename_files.sh prefix new_
# Example to add suffix: ./rename_files.sh suffix _old

# Check arguments
if [ $# -ne 2 ]; then
  echo "Usage: $0 prefix|suffix string"
  exit 1
fi

mode=$1
affix=$2

# Iterate over all files in the current directory
for file in *; do
  # Skip if not a file
  if [ ! -f "$file" ]; then
    continue
  fi

  if [ "$mode" = "prefix" ]; then
    newname="${affix}${file}"
  elif [ "$mode" = "suffix" ]; then
    # Extract filename without extension and extension part
    base="${file%.*}"
    ext="${file##*.}"
    if [ "$base" = "$file" ]; then
      # No extension
      newname="${file}${affix}"
    else
      newname="${base}${affix}.${ext}"
    fi
  else
    echo "Invalid mode: choose 'prefix' or 'suffix'"
    exit 1
  fi

  # Rename file
  mv -v "$file" "$newname"
done
~
~
~
~
~
~
```



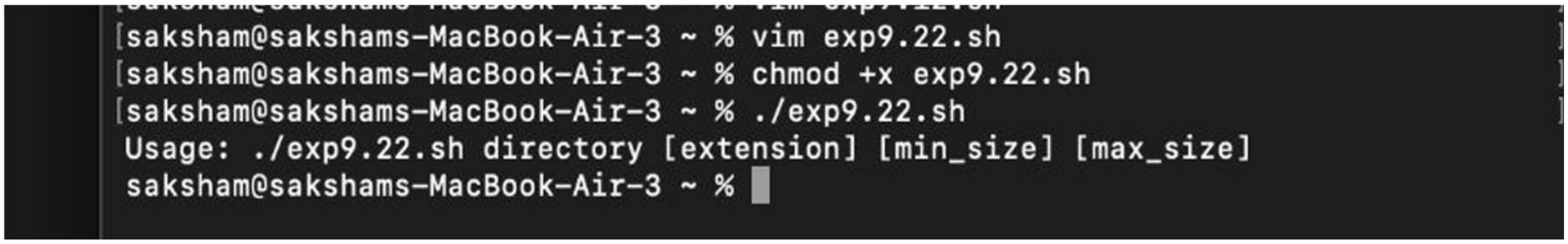
[redacted]

```
Last login: Tue Nov  4 10:20:43 on ttys002
saksham@sakshams-MacBook-Air-3 ~ % Vim exp9.12.sh
saksham@sakshams-MacBook-Air-3 ~ % chmod +x exp9.12.sh
saksham@sakshams-MacBook-Air-3 ~ % ./exp9.12.sh
Usage: ./exp9.12.sh prefix|suffix string
saksham@sakshams-MacBook-Air-3 ~ % ./exp9.12.sh new|old exp7.12.sh
zsh: command not found: old
saksham@sakshams-MacBook-Air-3 ~ % ./exp9.12.sh new|old exp 7.1.sh
zsh: command not found: old
saksham@sakshams-MacBook-Air-3 ~ %
```

2. Script that searches for files in a specified directory and its subdirectories, based on certain criteria like file extension or file size.

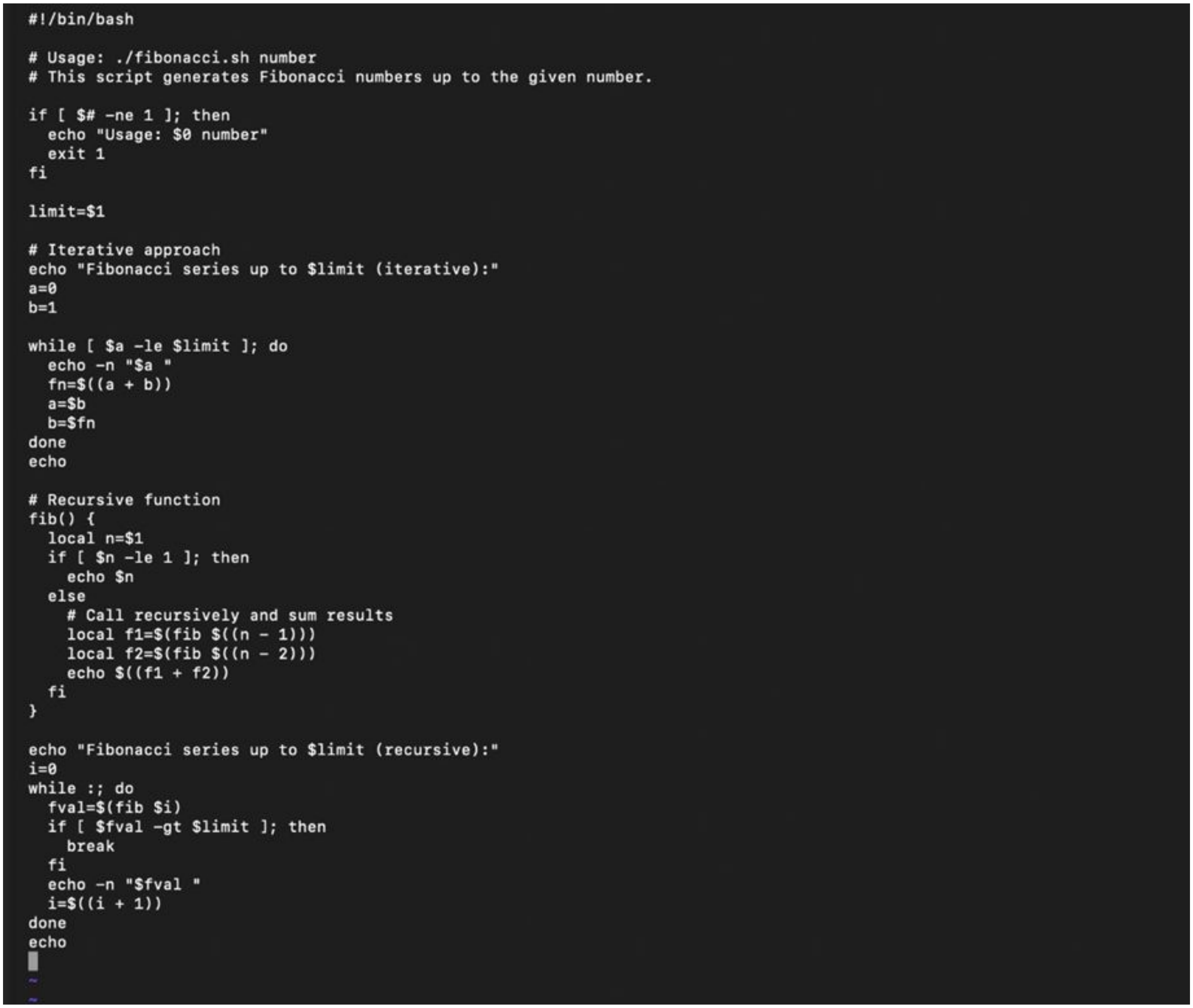


```
#!/bin/bash

# Usage: ./search_files.sh directory [extension] [min_size] [max_size]
# Example:
#   ./search_files.sh /path/to/dir txt       # Find .txt files in directory and subdirectories
#   ./search_files.sh /path/to/dir "" +100k  # Find files larger than 100 KB
#   ./search_files.sh /path/to/dir pdf 10k 1M # Find .pdf files between 10 KB and 1 MB

if [ $# -lt 1 ]; then
  echo "Usage: $0 directory [extension] [min_size] [max_size]"
  exit 1
fi

search_dir=$1
extension=$2
min_size=$3
max_size=$4

# Build the find command dynamically
cmd=(find "$search_dir" -type f)

# Filter by extension if provided
if [ -n "$extension" ]; then
  cmd+=(-name "*.$extension")
fi

# Filter by minimum size if provided (size suffixes: c=bytes, k=KB, M=MB, G=GB)
if [ -n "$min_size" ]; then
  # min_size with + means greater than that size
  cmd+=(-size "$min_size")
fi

# If max_size is provided, we filter files smaller than max_size by adding -size -max_size
# Min and max sizing in find are combined by chaining size filters
if [ -n "$max_size" ]; then
  cmd+=(-size -"${max_size}")
fi

# Execute the find command
"${cmd[@]}"

~
~
~
```

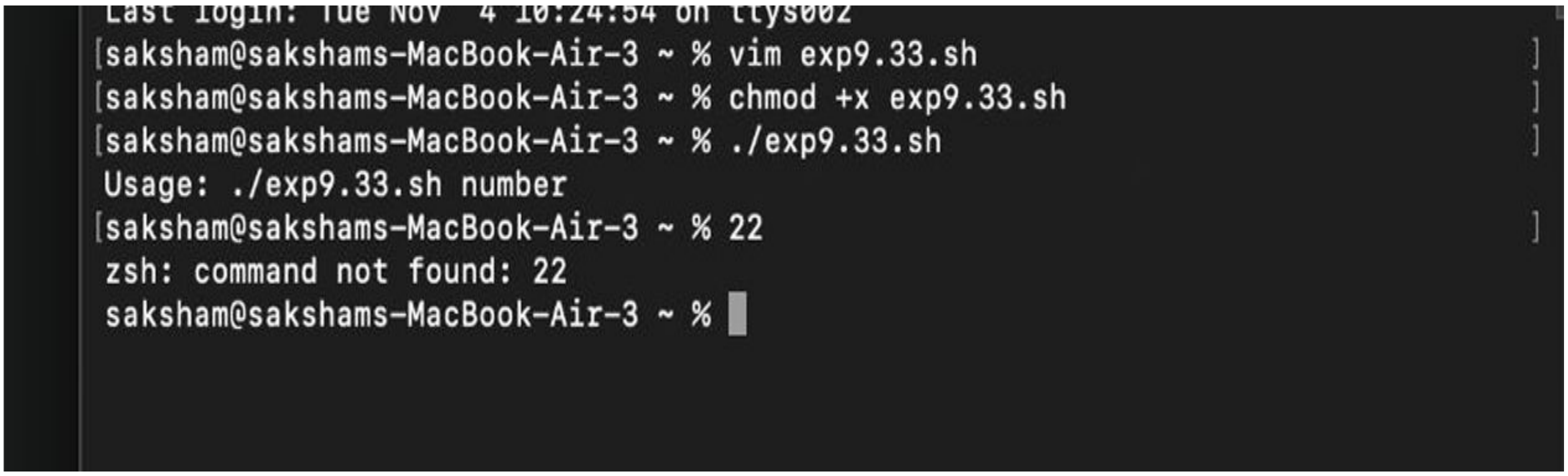


```
[saksham@sakshams-MacBook-Air-3 ~ % vim exp9.22.sh
[saksham@sakshams-MacBook-Air-3 ~ % chmod +x exp9.22.sh
[saksham@sakshams-MacBook-Air-3 ~ % ./exp9.22.sh
 Usage: ./exp9.22.sh directory [extension] [min_size] [max_size]
 saksham@sakshams-MacBook-Air-3 ~ %
```

3. Script that generates the Fibonacci series up to a given number, using loops or recursive functions.



```
#!/bin/bash

# Usage: ./fibonacci.sh number
# This script generates Fibonacci numbers up to the given number.

if [ $# -ne 1 ]; then
  echo "Usage: $0 number"
  exit 1
fi

limit=$1

# Iterative approach
echo "Fibonacci series up to $limit (iterative):"
a=0
b=1

while [ $a -le $limit ]; do
  echo -n "$a "
  fn=$((a + b))
  a=$b
  b=$fn
done
echo

# Recursive function
fib() {
  local n=$1
  if [ $n -le 1 ]; then
    echo $n
  else
    # Call recursively and sum results
    local f1=$(fib $((n - 1)))
    local f2=$(fib $((n - 2)))
    echo $((f1 + f2))
  fi
}

echo "Fibonacci series up to $limit (recursive):"
i=0
while :; do
  fval=$(fib $i)
  if [ $fval -gt $limit ]; then
    break
  fi
  echo -n "$fval "
  i=$((i + 1))
done
echo

~
~
```



```
Last login: Tue Nov  4 10:24:54 on ttys002
saksham@sakshams-MacBook-Air-3 ~ % vim exp9.33.sh
saksham@sakshams-MacBook-Air-3 ~ % chmod +x exp9.33.sh
saksham@sakshams-MacBook-Air-3 ~ % ./exp9.33.sh
Usage: ./exp9.33.sh number
saksham@sakshams-MacBook-Air-3 ~ % 22
zsh: command not found: 22
saksham@sakshams-MacBook-Air-3 ~ %
```

Learning outcomes

- Developed practical skill in automating file-management workflows using shell scripts, improving speed and reducing manual errors.
- Learned to manipulate filenames safely at scale, applying string operations and loops to handle many files in a directory
- Gained the ability to search files by attributes like extension and size, refining results using filters and conditional logic.
- Strengthened directory traversal strategies and filtering techniques to efficiently explore nested folder structures