OPERATING SYSTEM ASSIGNMENTS

Jadavpur University

Name: ARKA DAS

MCA 1st year 2nd Semester

Roll number: 002210503046

Session: 2022- 2024

Assignment: Set - 1

Set-1

Question – 1

Problem Statement:

Write a shell script which accepts length and breadth of a rectangle and calculates the area and perimeter of the rectangle.

Source Code:

```
isValid() {
      re='^[0-9]+([.][0-9]+)?$'
      if ! [[ $1 =~ $re && $2 =~ $re ]] ; then
         echo "error: Not a number"
         echo ""
         return 0
      else
      if [[ $(echo "$1 > 0" |bc -l) && $(echo "$2 > 0" |bc -l) ]] ; then
                  return 1
            else
                  echo "error: Invalid length"
                  echo ""
                  return 0
            fi
      fi
      return 1
}

main() {
      length=0
      breadth=0
      x=0
      while [ $x -ne 1 ]
      do
            echo "Enter length: "
            read length
            echo "Given length is: $length"
            echo "Enter breadth: "
            read breadth
            echo "Given breadth is: $breadth"
            isValid "$length" "$breadth"
            x=$?
      done
      area=`bc <<< "$length * $breadth"`
      temp=`bc <<< "$length + $breadth"`
      peri=`bc <<< "$temp * 2"`

      echo "Area of Rect: $area"
      echo "Perimeter of Rect: $peri"
}
main
```

Output:

```
arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_1$ bash assignment_1_1.sh
Enter length:
100
Given length is: 100
Enter breadth:
45
Given breadth is: 45
Area of Rect: 4500
Perimeter of Rect: 290
arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_1$ bash assignment_1_1.sh
Enter length:
ar23
Given length is: ar23
Enter breadth:
34df
Given breadth is: 34df
error: Not a number

Enter length:
10
Given length is: 10
Enter breadth:
23
Given breadth is: 23
Area of Rect: 230
Perimeter of Rect: 66
```

Question – 2

Problem Statement:

Write a shell script which accepts basic salary of an employee and calculates net salary and displays the salary slip.

Source Code:

```
isValid() {
      re='^[0-9]+([.][0-9]+)?$'
      if ! [[ $1 =~ $re ]] ; then
         echo "error: Not a number"
         echo ""
         return 0
      else
            if [[ $(echo "$1 > 0" |bc -l) ]] ; then
                  return 1
            else
                  echo "error: Invalid length"
                  echo ""
                  return 0
            fi
      fi
      return 1
}

main() {
      b_salary=0
      x=0
      while [ $x -ne 1 ]
      do
            echo "Enter basic salary: "
            read b_salary
            echo "Given basic salary: $b_salary"
            isValid "$b_salary"
            x=$?
      done

      da=0.04
      hra=0.06
      total=0
      da_amout=`bc <<< "$b_salary * $da"`
      hra_amout=`bc <<< "$b_salary * $hra"`
      total=`bc <<< "$b_salary + $da_amout + $hra_amout"`

      echo -e "Total salary: $total \t DA: $da_amout \t HRA: $hra_amout"
}

main
```

Output:

```
arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_1$ bash assignment_1_2.sh

Enter basic salary:

10500.344

Given basic salary: 10500.344

Total salary: 11550.377      DA: 420.013      HRA: 630.020

arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_1$ bash assignment_1_2.sh

Enter basic salary:

34sd

Given basic salary: 34sd

error: Not a number


Enter basic salary:

90

Given basic salary: 90

Total salary: 99.00     DA: 3.60    HRA: 5.40
```

Question – 3

Problem Statement:

Write a shell script which accepts a five digit number and prints sum of its digits.

Source Code:

```
isValid() {
      re='^[0-9]+$'
      if ! [[ $1 =~ $re ]] ; then
         echo "error: Not a Valid number"
         echo ""
         return 0
      else
            if [[ $(echo "$1 >= 0" |bc -l) ]] ; then
                  return 1
            else
                  echo "error: Invalid length"
                  echo ""
                  return 0
            fi
```

```
	fi
	return 1
}

main() {
	number=0
	x=0
	while [ $x -ne 1 ]
	do
		echo "Enter a number: "
		read number
		echo "Given number is: $number"
		isValid "$number"
		x=$?
	done

	sum=0
	while [ $number -gt 0 ]
	do
		rem=`expr $number % 10`
		sum=`expr $sum + $rem`
		number=`expr $number / 10`
	done

	echo "Sum of digits is: $sum"
}

main
```

Output:

```
arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_1$ bash assignment_1_3.sh
Enter a number:
12345
Given number is: 12345
Sum of digits is: 15
arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_1$ bash assignment_1_3.sh
Enter a number:
ser34
Given number is: ser34
error: Not a Valid number

Enter a number:
450
Given number is: 450
Sum of digits is: 9
```

Question – 4

Problem Statement:

Write a shell script which accepts a five digit number and prints the reverse number.

Source Code:

```
isValid() {
      re='^[0-9]+$'
      if ! [[ $1 =~ $re ]] ; then
         echo "error: Not a Valid number"
         echo ""
         return 0
      else
            if [[ $(echo "$1 >= 0" |bc -l) ]] ; then
                  return 1
            else
                  echo "error: Invalid length"
                  echo ""
                  return 0
            fi
      fi
      return 1
}

main() {
      number=0
      x=0
      while [ $x -ne 1 ]
      do
            echo "Enter a number: "
            read number
            echo "Given number is: $number"
            isValid "$number"
            x=$?
      done

      rev=0
      while [ $number -gt 0 ]
      do
            rem=`expr $number % 10`
            rev=`expr $rev \* 10`
            rev=`expr $rev + $rem`
            number=`expr $number / 10`
      done

      echo "Reversed number is: $rev"

}

main
```

Output:

```
arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_1$ bash assignment_1_4.sh
Enter a number:
12345
Given number is: 12345
Reversed number is: 54321
arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_1$ bash assignment_1_4.sh
Enter a number:
45ar34
Given number is: 45ar34
error: Not a Valid number

Enter a number:
5023
Given number is: 5023
Reversed number is: 3205
```

Question – 5

Problem Statement:

The /etc/passwd file stores user account information. It contains one entry per line for each user (user account) of the system. Each line contains seven fields which are separated by a colon (:) symbol. The fields are:

(i) Username

(ii) Password

(iii) User Id

(iv) Group Id

(v) User Id Info

(vi) Home Directory

(vii) Login Shell

Write a shell script which accepts a user login name and displays detail information about the users as available from the file /etc/passwd.

Source Code:

```
main() {
      if [ "$1" == "" ]
      then
            echo "No input given"
      else
            user=""
            line=""
            while [ "$user" == "" ]
            do
                  line=`cat /etc/passwd | grep -i $1`
                  user=`echo $line | cut -d ':' -f 1`
            done

            if [ "$user" == "" ]
            then
                  echo "User not found"

                  else
                        resutls=()
                        titles=("UserName", "Password", "UserId", "GroupId", "UserId
Info", "Home Directory", "Login Shell")

                        for i in {1..7}
                        do
                              info=`echo $line | cut -d ':' -f $i`
                              results+=($info)
                        done

                        for i in {0..6}
                        do
                              echo "${titles[$i]} : ${results[$i]}"
                        done
            fi

      fi

}

main $1
```

Output:

```
arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_1$ bash assignment_1_5.sh arka
UserName, : arka
Password, : x
UserId, : 1000
GroupId, : 1000
UserId Info, : arka,,,
Home Directory, : /home/arka
Login Shell : /bin/bash
arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_1$ bash assignment_1_5.sh user
UserName, : systemd-oom
Password, : x
UserId, : 108
GroupId, : 116
UserId Info, : systemd
Home Directory, : Userspace
Login Shell : OOM
arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_1$ bash assignment_1_5.sh
No input given
```

Assignment: Set - 2

Set-2

Question – 1

Problem Statement:

Write a shell script which, for all files in present directory displays whether it is a regular file or a directory.

Source Code:

```
main() {

      line=`ls -F`

      for i in `echo $line`
      do
            echo -n $i
            length=`echo $i | wc -c`
            length=`expr $length - 1`
            #echo $length
            c=`echo $i | cut -c $length`
            if [ $c == "/" ]
                  then
                        echo -n " -> "
                        echo "Directory"
                  else
                        echo -n " -> "
                        echo "File"
            fi
      done
}

main > out_q_1.txt
cat out_q_1.txt
```

Output:

```
arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_2$ bash q_1.sh

folder/ -> Directory

hello.txt -> File

out_q_1.txt -> File

out_q_2.txt -> File

out_q_3.txt -> File

out_q_4.txt -> File

out_q_5.txt -> File

q_1.sh -> File

q_2.sh -> File
```

```
q_3.sh -> File

q_4.sh -> File

q_5.sh -> File

temp/ -> Directory
```

Question – 2

Problem Statement:

The PATH variable is an environment variable that contains an ordered list of paths that Linux will search for executables when running a command. Write a shell script to display all the directories in the PATH variable in a simple way, i.e., one line per directory. In addition, display information about each directory, such as the permissions and the modification times.

Source Code:

```
main() {
      paths=`echo $PATH`

      for i in $(echo $paths | tr ":" "\n")
      do
            #echo $i
            if [ -d "$i" ]
            then
                  info=`ls -ld $i`
                  #echo $info
                  d1=`echo $info | cut -d " " -f 1`
                  d2=`echo $info | cut -d " " -f 6`
                  d3=`echo $info | cut -d " " -f 7`
                  d4=`echo $info | cut -d " " -f 8`
                  echo "'$i' -> $d1 $d2 $d3 $d4"
            else
                  echo "'$i' -> does not exist"
            fi

      done
}

main > out_q_2.txt
cat out_q_2.txt
```

Output:

```
arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_2$ bash q_2.sh
'/usr/local/sbin' -> drwxr-xr-x Feb 23 09:27
'/usr/local/bin' -> drwxr-xr-x Feb 23 09:27
'/usr/sbin' -> drwxr-xr-x Jun 2 21:42
'/usr/bin' -> drwxr-xr-x Jun 6 07:59
```

```
'/sbin' -> lrwxrwxrwx Apr 2 11:49
'/bin' -> lrwxrwxrwx Apr 2 11:49
'/usr/games' -> drwxr-xr-x Feb 23 09:28
'/usr/local/games' -> drwxr-xr-x Feb 23 09:27
'/snap/bin' -> drwxr-xr-x Jun 3 09:19
'/snap/bin' -> drwxr-xr-x Jun 3 09:19
```

Question – 3

Problem Statement:

Write a shell script which displays vendor id, model name, CPU MHz, cache size information about the processor present in your computer. Hint: most of this information can be obtained by reading the file /proc/cpuinfo.

Source Code:

```
main() {
      title=("vendor_id" "model name" "cpu MHz" "cache size")

      for i in {0..3}
      do
            v=`cat /proc/cpuinfo | grep "${title[$i]}"`
            echo "$v"
      done
}

main > out_q_3.txt
cat out_q_3.txt
```

Output:

```
arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_2$ bash q_3.sh

vendor_id   : AuthenticAMD

vendor_id   : AuthenticAMD

vendor_id   : AuthenticAMD

vendor_id   : AuthenticAMD
```

```
model name  : AMD Ryzen 5 2500U with Radeon Vega Mobile Gfx

model name  : AMD Ryzen 5 2500U with Radeon Vega Mobile Gfx

model name  : AMD Ryzen 5 2500U with Radeon Vega Mobile Gfx

model name  : AMD Ryzen 5 2500U with Radeon Vega Mobile Gfx

cpu MHz          : 1996.191

cpu MHz          : 1996.191

cpu MHz          : 1996.191

cpu MHz          : 1996.191

cache size  : 512 KB

cache size  : 512 KB

cache size  : 512 KB

cache size  : 512 KB
```

Question – 4

Problem Statement:

Write a shell script to show your home directory, Operating System type, version, release number, kernel version and current path setting. Hint: use uname command or use content of /proc/sys/kernel/osrelease file.

Source Code:

```
main() {
      title=("OS type" "version" "release number" "kernel version")
      uname_flags=("o" "v" "r" "v")

      homeDir=`ls /home`
      echo "Home directory :" $homeDir

      for i in {0..3}
      do
            info=`uname -${uname_flags[$i]}`
            echo "${title[$i]} : $info"
      done

      echo "All Path settings :" $PATH
}

main > out_q_4.txt
cat out_q_4.txt
```

Output:

```
arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_2$ bash q_4.sh

Home directory : arka

OS type : GNU/Linux

version : #44~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Mon May 22 13:39:36 UTC 2

release number : 5.19.0-43-generic

kernel version : #44~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Mon May 22 13:39:36 UTC 2

All Path settings :
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/ga
mes:/snap/bin:/snap/bin
```

Question – 5

Problem Statement:

Write a shell script to display a summary of the disk space usage for each directory argument (and any subdirectories), both in terms of bytes, and kilobytes or megabytes (whichever is appropriate). [du -b]

Source Code:

```
main(){
      for dir in $@
      do
            if [[ -d $dir ]]
            then
                  echo "Directory name: $dir"
                  a=`du -s -b "$dir" | cut -f 1`
                  echo "Size in bytes: $a"
                  b=`du -s -h "$dir" | cut -f 1`
                  echo "Size in KB: $b"
            else
                  echo "Folder does not exist"
            fi
      done
}

main $@ > out_q_5.txt
cat out_q_5.txt
```

Output:

```
arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_2$ bash q_5.sh folder

Directory name: folder

Size in bytes: 7549

Size in KB: 8.0K

arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_2$ bash q_5.sh temp

Directory name: temp

Size in bytes: 4096

Size in KB: 4.0K

arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_2$ bash q_5.sh hello

Folder does not exist
```

Assignment: Set - 3

Set-3

Question – 1

Problem Statement:

Write a shell script which reads an input file that contains three integers in each line. The script should display the sum of all integers in each line.

Source Code:

```
isValid() {
      re='^[0-9]+([.][0-9]+)?$'
      if ! [[ $1 =~ $re ]] ; then
         return 0
      fi
      return 1
}

main() {
      sum=0
      i=1
      while read -r line
      do
            currSum=0
            a=0
            b=0
            c=0

            a=`echo $line | cut -d ' ' -f 1`
            isValid "$a"
            x=$?
            if [[ $x -ne 1 ]]; then
                  echo "contents not valid"
                  exit
                  a=0
            fi

            b=`echo $line | cut -d ' ' -f 2`
            isValid "$a"
            x=$?
            if [[ $x -ne 1 ]]; then
                  echo "contents not valid"
                  exit
                  b=0
            fi

            c=`echo $line | cut -d ' ' -f 3`
            isValid "$a"
            x=$?
            if [[ $x -ne 1 ]]; then
                  echo "contents not valid"
                  exit
                  c=0
            fi
            currSum=$(($a + $b + $c))
            echo "Sum in line:$i is: $currSum"
            sum=$(($sum + $currSum))
            i=`expr $i + 1`
      done
      echo "Total sum is: $sum"
}

main < $1
```

Output:

```
file contents
1sd 2 3
4 5 6
7sdf 8 9

arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_3$ bash q_1.sh
Sum in line:1 is: 5
Sum in line:2 is: 15
Sum in line:3 is: 17
Total sum is: 37
```

Question – 2

Problem Statement:

Write a shell script to find out how many file and directory are there in the current directory. Also list the file and directory names separately.

Source Code:

```
main() {
      fileCount=`ls -l | grep ^- | wc -l`
      dirCount=`ls -l | grep ^d | wc -l`
      echo "Total number of directories: $dirCount"
      echo "Total number of files: $fileCount"
      echo ""

      line=`ls -F`
      for i in `echo $line`
      do
            echo -n $i
            length=`echo $i | wc -c`
            length=`expr $length - 1`
            c=`echo $i | cut -c $length`
            if [ $c == "/" ]
                  then
```

```
                        echo -n " -> "
                        echo "Directory"
                  else
                        echo -n " -> "
                        echo "File"
            fi
      done
}

main > out_q_2.txt
cat out_q_2.txt
```

Output:

```
arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_3$ bash q_2.sh

Total number of directories: 3

Total number of files: 12

emptyFolder/ -> Directory

exeToshell.sh -> File

input_1.txt -> File

out_q_1.txt -> File

out_q_2.txt -> File

out_q_3.txt -> File

out_q_6.txt -> File

q_1.sh -> File

q_2.sh -> File

q_3.sh -> File

q_4.sh -> File

q_5.sh -> File

q_6.sh -> File

temp/ -> Directory

testFolder/ -> Directory
```

Question – 3

Problem Statement:

Write a script that adds up the sizes reported by the ls command for the files in the current directory. The script should print out only the total number of bytes used.

Source Code:

```
main() {
      totalSize=0
      for file in *
      do
            if [ -f "$file" ]
            then
                  size=`ls -l "$file" | cut -d ' ' -f5`
                  totalSize=$(($totalSize + $size))
            fi
      done
      echo "Total number of bytes used by files is: $totalSize"
}

main > out_q_3.txt
cat out_q_3.txt
```

Output:

```
arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_3$ bash q_3.sh

Total number of bytes used by files is: 2276
```

Question – 4

Problem Statement:

Write a shell scripts that delete all temporary files (end with ~) in current directory.

Source Code:

```
#!/bin/bash
pwd
ls
rm *~
echo "After deletion"
ls
pwd
```

Output:

```
arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_3$ bash q_4.sh

emptyFolder    out_q_1.txt  out_q_6.txt  q_3.sh  q_6.sh  temp

exeToshell.sh  out_q_2.txt  q_1.sh q_4.sh  t1~      testfile~

input_1.txt    out_q_3.txt  q_2.sh q_5.sh  t2~      testFolder

After deletion

emptyFolder    input_1.txt  out_q_2.txt  out_q_6.txt  q_2.sh  q_4.sh  q_6.sh
testFolder

exeToshell.sh  out_q_1.txt  out_q_3.txt  q_1.sh       q_3.sh  q_5.sh  temp
```

Question – 5

Problem Statement:

Write a shell script to rename file having extension .sh to .exe.

Source Code:

```
for file in *.sh
do
     mv -- "$file" "${file%.sh}.exe"

done
```

Output:

```
arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_3/testFolder$ ls

exeToshell.sh  out_q_6.txt  q_2.sh  q_3.sh  q_5.sh  q_6.sh

arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_3/testFolder$ bash q_5.sh

arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_3/testFolder$ ls

exeToshell.exe  out_q_6.txt  q_2.exe  q_3.exe  q_5.exe  q_6.exe
```

Question – 6

Problem Statement:

Write a shell script to rename file having extension .sh to .exe.

Source Code:

```
main() {
      count=0
      for file in *.sh
      do
            if [ -f "$file" ]
            then
                  count=$(($count + 1))
            fi
      done
      echo "Total number of shell files is: $count";
}

main > out_q_6.txt
cat out_q_6.txt
```

Output:

```
arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_3$ ls -l

-rwxrwx--- 1 arka arka   60 May 16 07:48 exeToshell.sh

-rwxrwx--- 1 arka arka   23 May 16 08:05 input_1.txt

-rwxrwx--- 1 arka arka   79 Jun  6 20:03 out_q_1.txt

-rwxrwx--- 1 arka arka  340 Jun  6 20:04 out_q_2.txt

-rwxrwx--- 1 arka arka   45 Jun  6 20:06 out_q_3.txt

-rwxrwx--- 1 arka arka   34 May 16 08:01 out_q_6.txt

-rwxrwx--- 1 arka arka  708 May 16 07:36 q_1.sh

-rwxrwx--- 1 arka arka  482 Apr 25 07:05 q_2.sh

-rwxrwx--- 1 arka arka  254 May  8 06:39 q_3.sh

-rwxrwx--- 1 arka arka   46 May 16 07:45 q_4.sh

-rwxrwx--- 1 arka arka   61 May 16 07:44 q_5.sh

-rwxrwx--- 1 arka arka  189 May 16 08:01 q_6.sh

arka@Ubuntu22:~/Desktop/Shell_scripts/custom/set_3$ bash q_6.sh

Total number of shell files is: 7
```

Complete menu-driven program for all the questions of set – 1, 2 and 3

Problem Statement:

Write a menu-driven shell script which contains all the questions of set 1, 2 and 3. The program should display a list of sets first and then any set can be selected. From the selected set, all the questions will be now displayed. The user selects which code to run. Once done the code must return to the previous level to select set state.

Source Code:

```
main() {

      setChoice=0
      qChoice=0
      while true
      do
            echo "1 -> set_1"
            echo "2 -> set_2"
            echo "3 -> set_3"
            echo "4 -> Exit"
            echo "Enter your choice: "
            read setChoice

            if [ $setChoice -lt 1 -a $setChoice -gt 4 ]
            then
                  echo "Wrong choice"
            fi

            case $setChoice in
            1)
                  cd ./set_1
                  while true
                  do
                        progName="assignment_1_"
                        echo "0 -> go to set choice"
                        echo "1 -> question_1"
                        echo "2 -> question_2"
                        echo "3 -> question_3"
                        echo "4 -> question_4"
                        echo "5 -> question_5"
                        read qChoice
                        if [ $qChoice -eq 0 ]
                        then
                              cd $OLDPWD
                              break
                        fi
                        progName="${progName}${qChoice}"
                        progName="${progName}".sh""
                        if [ $qChoice -eq 5 ]
                        then
                              echo "Enter user name to find: "
                              read userName
                              bash $progName "$userName"
                        else
```

```
                bash $progName
            fi
        done
    ;;

    2)
        cd ./set_2
        while true
        do
            progName="q_"
            echo "0 -> go to set choice"
            echo "1 -> question_1"
            echo "2 -> question_2"
            echo "3 -> question_3"
            echo "4 -> question_4"
            echo "5 -> question_5"
            read qChoice
            if [ $qChoice -eq 0 ]
            then
                cd $OLDPWD
                break
            fi
            progName="${progName}${qChoice}"
            progName="${progName}".sh""
            if [ $qChoice -eq 5 ]
            then
                echo "Enter folder name to find: "
                read folderName
                bash $progName "$folderName"
            else
                bash $progName
            fi
        done
    ;;

    3)
        cd ./set_3
        while true
        do
            progName="q_"
            echo "0 -> go to set choice"
            echo "1 -> question_1"
            echo "2 -> question_2"
            echo "3 -> question_3"
            echo "4 -> question_4"
            echo "5 -> question_5"
            echo "6 -> question_6"
            read qChoice
            if [ $qChoice -eq 0 ]
            then
                cd $OLDPWD
                break
            fi
            progName="${progName}${qChoice}"
            progName="${progName}".sh""

            if [ $qChoice -eq 1 ]
            then
```

```
                              echo "Enter target filename: "
                              read fileName
                              bash $progName "$fileName"

                         elif [ $qChoice -le 6 -a $qChoice -ge 4 ]
                         then
                              echo "Enter target folder: "
                              read folderName
                              cd ./$folderName
                              bash $progName "$folderName"
                              cd $OLDPWD
                         else
                              bash $progName
                         fi
                    done
               ;;

               4)
                    break
               esac


          done

     }

     main $@
```

Assignment: Set - 4

Set-4

Question – 1

Problem Statement:

Write a C program to create a child process. The parent process must wait until the child finishes. Both the processes must print their own pid and parent pid. Additionally the parent process should print the exit status of the child.

Source Code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

int main() {
    pid_t pid;
    int status;

    pid = fork();

    if (pid < 0) {
        perror("Fork failed");
        exit(1);
    }
    else if (pid == 0) {
        printf("Child process - PID: %d, Parent PID: %d\n", getpid(), getppid());
        sleep(1);
        exit(50);
    }
    else {
        // Parent process
        printf("Parent process - PID: %d, Child PID: %d\n", getpid(), pid);
        wait(&status);
        printf("Child process exited with status: %d\n", WEXITSTATUS(status));
    }

    return 0;
}
```

Output:

```
arka@Ubuntu22:~/Desktop/Shell_scripts/set_4$ gcc q_1.c

arka@Ubuntu22:~/Desktop/Shell_scripts/set_4$ ./a.out

Parent process - PID: 5205, Child PID: 5206

Child process - PID: 5206, Parent PID: 5205

Child process exited with status: 50
```

Question – 2

Problem Statement:

Write a C program which prints prime numbers between the range 1 to 10,00,000 by creating ten child processes and subdividing the task equally among all child processes, i.e., the first child should print prime numbers in the range 1 to 1,00,000, the second child in the range 1,00,001 to 2,00,000, ... The child processes must run in parallel and the parent process must wait until all the child processes finish.

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdbool.h>
#include <math.h>

#define RANGE 100000
#define NUM_CHILDREN 10

bool is_prime(int number) {
    if (number < 2)
        return false;

    int sqrt_num = sqrt(number);
    for (int i = 2; i <= sqrt_num; i++) {
        if (number % i == 0)
            return false;
    }

    return true;
}

void print_primes(int start, int end) {
    for (int number = start; number <= end; number++) {
        if (is_prime(number))
            printf("%d ", number);
    }
    printf("\n");
}

int main() {
    pid_t pid;
    int status;

    for (int i = 0; i < NUM_CHILDREN; i++) {
        pid = fork(); // Create a child process

        if (pid < 0) {
            perror("Fork failed");
            exit(1);
        } else if (pid == 0) {
            // Child process
            int start = (i * RANGE) + 1;
            int end = (i + 1) * RANGE;

            printf("Child process %d - PID: %d, Parent PID: %d\n", i+1, getpid(),
getppid());
            printf("Printing prime numbers between %d and %d:\n", start, end);
            print_primes(start, end);

            exit(0); // Exit the child process
        }
    }

    // Parent process
    for (int i = 0; i < NUM_CHILDREN; i++) {
        wait(&status); // Wait for each child process to finish
        printf("Child process %d finished.\n", i+1);
    }

    return 0;
}
```

Output:

Question – 3

Problem Statement:

Write a C program which creates a child process. The parent process sends a string (input by user) which the child process inspects and sends ”YES” back to the parent if the string is a palindrome, otherwise it sends ”NO”. The IPC to be used is pipe. Both the processes terminate when the input string is “quit”.

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>
#define BUFFERSIZE 50

int isPalindrome(char* data);


int main()
{
      char buffer[BUFFERSIZE] = " ";
```

```
	int pipe_1[2];	//0 -> read , 1 -> write
	int pipe_2[2];

	if(pipe(pipe_1) == -1)
		return 0;
	if(pipe(pipe_2) == -1)
		return 0;

	pid_t pid = fork();
	if(pid == -1)
		return 0;
	if(pid == 0) {
		//child
		close(pipe_1[0]); // close read from pipe 1
	  close(pipe_2[1]); // close write from pipe 2

	  while(1) {
	      read(pipe_2[0], buffer, BUFFERSIZE);
	      if (strcmp(buffer, "quit") == 0)
	          break;

	      if(isPalindrome(buffer))
	            write(pipe_1[1], "Yes", 4);
	      else
	            write(pipe_1[1], "No", 3);
	  }

	  close(pipe_1[1]);
	  close(pipe_2[0]);
	  exit(20);
	}
	else {
		//parent
		close(pipe_1[1]); // close write from pipe 1
	  close(pipe_2[0]); // close read from pipe 2

	  while(1) {
	      printf("\nEnter a string: ");
	      fgets(buffer, BUFFERSIZE, stdin);
	      buffer[strcspn(buffer, "\n")] = '\0'; // make \n and make it null
	      //fputs(buffer, stdout);

	      if (strcmp(buffer, "quit") == 0) {
	          write(pipe_2[1], buffer, BUFFERSIZE);
	          break;
	       }

	       write(pipe_2[1], buffer, BUFFERSIZE);
	       read(pipe_1[0], buffer, BUFFERSIZE);
	       fputs(buffer, stdout);
	  }

	}


}
```

```
int isPalindrome(char data[]) {
      int i, len;
    len = strlen(data);

    for (i = 0; i < len / 2; i++) {
        if (data[i] != data[len - i - 1])
            return 0;
    }
    return 1;
}
```

Output:

```
arka@Ubuntu22:~/Desktop/Shell_scripts/set_4$ gcc q_3.c

arka@Ubuntu22:~/Desktop/Shell_scripts/set_4$ ./a.out


Enter a string: hello

No

Enter a string: dad

Yes

Enter a string: yess

No

Enter a string: test

No

Enter a string: madam

Yes

Enter a string: quit
```

Question – 4

Problem Statement:

Write a C program which prints the following menu

1. ls

2. pwd

3. uname

4. exit

When, the user provides an input, the parent process creates a child process [if user's choice is between 1-3] and executes the corresponding command [use execv() system call]. The main process waits for the child to finish and displays the menu again. The parent process terminates if user's choice is 4.

Source Code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int main()
{
      int choice = 0;
      while(1) {
            printf("1. ls\n");
        printf("2. pwd\n");
        printf("3. uname\n");
        printf("4. exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        if(choice >= 1 && choice <= 3) {
           pid_t pid = fork();
           if(pid < 0) {
                 printf("\nFork failed");
                 exit(1);
           }
           else if(pid == 0) {
                 //child
                 char *command;
                 switch(choice) {
                 case 1:
                       command = "ls";
                       break;
                 case 2:
                       command = "pwd";
                       break;
                 case 3:
                       command = "uname";
                       break;
                 }
```

```
                execlp(command, command, NULL);
                printf("error");
                exit(EXIT_FAILURE);
            }
            else {
                //parent
                int status;
               waitpid(pid, &status, 0);
               printf("\n");
            }

        }
            else if(choice == 4) {
                return 0;
            }
            else {
                printf("\nInvalid option\n");
                continue;
            }

      }
}
```

Output:

```
arka@Ubuntu22:~/Desktop/Shell_scripts/set_4$ gcc q_4.c

arka@Ubuntu22:~/Desktop/Shell_scripts/set_4$ ./a.out

1. ls

2. pwd

3. uname

4. exit

Enter your choice: 1

a.out  prog3.c   prog4.c  q_1.c   q_2.c  q_3.c  q_4.c


1. ls

2. pwd

3. uname

4. exit

Enter your choice: 2

/home/arka/Desktop/Shell_scripts/set_4


1. ls
```

```
2. pwd
3. uname
4. exit
Enter your choice: 3
Linux

1. ls
2. pwd
3. uname
4. exit
Enter your choice: 7

Invalid option
1. ls
2. pwd
3. uname
4. exit
Enter your choice: 4
```