CS29206: Systems Programming Lab
Spring 2025
Test 2, April 3, 2025

Time: 1 Hour **Full marks: 50**

ROLL NO.: ____________________ **NAME:** ________________________________

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	TOTAL
Q9	**Q10**	**Q11**						

1. Suppose your current directory has a subdirectory named “users”. The subdirectory “users” may have more subdirectories under it, which may again have more subdirectories under them and so on. Show the Linux command (single command to be run from the $ prompt in your current directory) to show all details of all files in all directories under the subdirectory “users” (including it). (2)

   [ANS] *ls –lR ./users*

2. Suppose that you have a large file named *myfile.c* where you may have a large number of unused variables (variables declared but never assigned or referenced). Also, you may have called a lot of functions without first declaring either the prototype or writing the body of the function before the call. The file compiles with gcc without any errors. However, you want to get warnings for both of the above so that you may go clean up unused variables and declare function prototypes properly. Show the gcc command (single command to be executed at $ prompt) for doing this. You should not get warnings for anything else. Assume that gcc by default will give no warnings. (2)

   [ANS] $gcc –Wunused -Wimplicit myfile.c

3. Consider the gcc command below. If the file *myfile.c* prints the value of A*B/C, what will be printed? Justify your answer in one sentence only. (2)

```
$gcc -Wall -DA=″2 + 3″ -DB=″8 - 6″ -DC=″3 + 4″
```

   [ANS] *28, because values of A, B, and C will be literally substituted (and not evaluated before substitution) and then the expression will be evaluated as per standard precedence rules.*

4. Suppose your current directory is X, with two subdirectories Y and Z under it. X has a file *x.c*, Y has a file *y.c*, and Z has a file *z.c*. Y and Z each have a makefile that has (among other things) a target named "clean" to clean all .o files in that directory. X has a makefile with target clean as below:

```
clean:
          cd Y
          make clean
          cd Z
          make clean
          rm *.o
```

   If all .c files in directory X and sub-directories Y, Z have been compiled already into .o files, how many total .o files will be there in the directory X, sub-directory Y and sub-directory Z after the following command is executed from the directory X? Justify your answer in 1-2 sentence max. (2)

```
$make clean
```

   [ANS] Both of the following are accepted.

   X = 1 , Y = 1 , Z = 1

   Justification: Since the cd and make are not done in the same line, make is executed in a different shell than cd. So after "cd Y", the "make clean" executes the makefile in X directory itself again, which goes in an infinite loop and nothing is deleted from any of the three directories

   X = 0 , Y = 1 , Z = 1

   Justification: As stated above, the .o files in X are also not deleted, but given marks even if that is missed and said that the rm will delete the .o files in X, as long as the reasoning with respect to the shells is correct.

5. Suppose you want to search for all lines in a text file *myfile.txt* that contains any of the strings "systems" or "program" (occurring as substring is ok, but search should be case insensitive) but does not contain any digits (0 to 9). The result of the search should be stored in a file called output.txt. Write a single line command that will do this when executed from the $ prompt. (3)

   [ANS] *grep –i –e "systems" –e "program" myfile.txt | grep –v '[0-9]' > output.txt*

   (Other variations are possible and have been given due credit if correct)

6. A software package is written in C programming language. The package is a huge codebase with many directories and sub-directories under a root directory containing total 364 files, creating many executable files after compilation. What command (with full options) will you write at the $ prompt in the root directory to list all the file names and line numbers (with its content) in them where main function (main()) is written? Remember C programming language is a case-sensitive language.

Assume that no main function in the codebase takes any arguments and that there are no unnecessary spaces used in the word main() anywhere. (3)

[ANS] *grep -R -n -w "main()" ./*

7. Consider a text file result.txt containing student grades. Each line in the file contains the grades of one student. Each line has the following fields: <first name>;<last name>;<marks in Maths>;<marks in Physics>;<marks in Chemistry>. The fields are separated by semicolon (;). A student gets an EX in a course if he/she gets ≥ 90 marks in the course. Write a single awk command (to be executed from the $ prompt) to print the first and last names (all in uppercase and separated by space) of all students who gets EX in all of the three courses. (3)

   [ANS]
   $gawk –F';' '{if (($3 >= 90) && ($4 >= 90) && ($5 >= 90)) {print toupper($1) " " toupper($2)}}' result.txt

   (Some other variations are possible and have been given due credit if correct)

8. Consider a directory containing the following source files:

prime.c	/* contains a function that will be used in mainPrime.c */
mainPrime.c	/* main() is in this file */
prime.h	/* The header file is included in mainPrime.c source file.*/
Makefile	/* Make file */

Fill in the blanks (*one blank line may indicate one or multi-line entry*), if we execute the following commands from the $ prompt. (8)
NOTE: There will be negative marking for each extra (unwanted) command.

```
$ make clean

rm a.out mainPrime.o libprime.a prime.o

$ make

gcc -Wall -c mainPrime.c

gcc -Wall -c prime.c

ar -rcs libprime.a prime.o

gcc mainPrime.o -L. -lprime

$ touch prime.c

$ make
```

```
gcc -Wall -c prime.c

ar -rcs libprime.a prime.o

gcc mainPrime.o -L. -lprime

$
```

9. Write a single shell command to be directly executed from the $ prompt to declare a 4-element read-only array A and initialize it to the integers (37, 51, 23, 10). (3)

   [ANS] *declare –air A=([0]=37 [1]=51 [2]=23 [3]=10)*

   (Other variations are possible and have been given due credit if correct)

10. In last year's campus placements, the following 8 companies approached CDC first. For each company, CDC stores the following information in a file – serial number, name of the company, per person CTC, number of available position, and the country for which they are hiring. The fields in each line are separated by '|' as shown below. While preparing the list, the person enter comma as per Indian convention – 1,10,00,000 indicates 1 crore 10 lakhs only.

    ```
    1|Google|1,10,00,000|2|USA

    2|Adobe|60,00,000|10|India

    3|Amazon|1,20,00,000|3|USA

    4|Flipkart|1,00,00,000|4|India

    5|Facebook|1,05,00,000|1|USA

    6|Intel India|50,00,000|5|India

    7|Microsoft USA|1,10,00,000|2|USA

    8|Intel UK|90,00,000|4|UK
    ```

    CDC decided that the company who is spending most for our institute (= per person CTC multiplied by the number of available positions), will be given the top priority. A second year CS student wrote the following awk code which, given the above data, prints

    Adobe hires for India and spends the most.

    Fill in the blanks below to complete the awk code so that it is generalized to handle any number of company details with the above format. (12)

    ```
    #!/usr/bin/gawk -f
    function CTC (a)
    ```

```
{
        n=split(a,arr,",")
        ctc=arr[n]
        mult=10
        for(i=n-1;i>0;i--) {
                mult*=100
                ctc+=arr[i]*mult
        }
        return ctc
}
BEGIN {
        FS = "|"
        slNo=0
        maxCTC=-99
}
{
        slNo+=1
        company[slNo]=$2
        grossCTC=$4*CTC($3)
        country[slNo]=$5
        if(maxCTC<grossCTC) {
                maxCTC=grossCTC
                bestOne=slNo
        }
}
END {
        print  company[bestOne] " hires for “
country[bestOne] " and spends the most."
}
```

11. Fill in the blanks such that the shell program that takes a list of file extensions (strings) to search for as command line arguments, will print all regular files in the current directory that end with any of those extensions. [10]

```
#!/bin/bash
declare -A exts
declare -A list
```

```
for i in $@; do
        exts[$i]=$i
done
for val in  ${!exts[*]}; do
        list[$val]=$(ls | grep -e $val$)
        for name in ${list[$val]}; do
                if [ -f $name ]; then
                    echo $name
                fi
        done
done
```

Other variations are possible are possible and have been given due credit if correct.