Winter Semester 2019, Final Exam II
Computer Practicum I (CS/BF)

Rules:

- You have 120 minutes to complete this exam.
- This test is open-book/notes. You may use any paper resources.
- You may not use any computing devices, USB devices or cell phones.
- If you enter the room, you must turn in an exam and will not be permitted to leave without doing so.

Question 1) (20 points) (Bash commands)

I. Use the information displayed below to answer **questions 1 to 3:**

```
$ cat students.txt
John Mark  79 60
Brhn Mans 65 45
John Maitt 78 102
Lilly Nova 83 15
Martin Scott 80 50
Mariana Shern 81 30
Sheldon Park 84 10
Austin Holmes 62 85
Ivy Cooper 87 25
```

1. Write a single command to display the first 5 data rows of the given information text.
2. Write a single command to get the following output.

```
hn Ma
hn Ma
hn Ma
lly N
rtin
Riana
Eldon
stin
y Coo
```

3. Write a single command to count the number of characters in the given information text.

II. For **questions 4 to 6,** each question must be answered with a **single Unix command,** without making use of the command separator character ; (semicolon). You can assume that all files are contained in your current directory (unless specified).

4. Write a single Unix command to save the current disk usage into a file called current_usage.txt.
5. Write a single Unix pipeline command to store a **sorted list of directories** which have **all read,write and execute permission** to the **user** in your current working directory and write the output results to a file called "Directories.txt"
6. Write a single Unix pipeline command to display only lines 5 to 10 in your "Directories.txt" file.

III. Redirect your terminal **history** in to the **Question1.txt**

Question 2) Write a Bash shell script named **DNA.sh** that prompts the user to enter a DNA sequence **character by character**. The length of the DNA sequence should be provided as a **command line argument. (20 points)** (Bash script)

There are four potential base characters that exist in DNA sequences: "A", "T", "G", and "C".
"A" and "T" are always paired together, and "G" and "C" are always paired together.

- Validate whether the length of the DNA Sequence is provided as command line argument, if not output **"Please provide the length of the DNA sequence!"**
- If the character is "A" , then it needs to be paired with "T" to give the output "A-T".
- If the character is "T", then it needs to be paired with "A" to give the output "T-A".
- If the character is "C" , then it needs to be paired with "G" to give the output "C-G".
- If the character is "G", then it needs to be paired with "C" to give the output "G-C".
- Else give the out output as **"X-This is not a DNA base character"**

Sample Run #1: **voidsky$ ./DNA.sh 3**
Enter DNA base 1: A
Pair: A-T
Enter DNA base 2: G
Pair: G-C
Enter DNA base 3: T
Pair: T-A

Sample Run #2: **voidsky$ ./DNA.sh 2**
Enter DNA base 1: A
Pair: A-T
Enter DNA base 2: B
Pair: B-This is not a DNA base character

Sample Run #3: **voidsky$ ./DNA.sh**
Please provide the length of the DNA sequence!

Question 3) Give an implementation of the standard C library function strcat, strcpy The basic idea of this function is to add to a string by appending the contents of another string. **(25 points)** (C basics) The size of the string is not fixed. User should be able to concatenate strings with any length
Example:

```
char s[20];
strnSize(s) // should return the length of a string
strncpy(s, "first name", 20);
strncat(s, " last name", 5);
printf("%s\n", s);
```

prints first name last name.

Null character '\0' should be present at the end of each string
The full specification of strncat is: char* strncat(char *dest, char *src, int n); • Append up to n characters from the contents of src to the end of dest.
• If the null ('\0') character that terminates src is encountered before n characters have been copied, then the null character is copied but no more.
• If no null character appears among the first n characters of src, then the first n characters of src are copied and a null character is supplied to terminate dest, i.e., n+1 characters in all are written.

• If n ≤ 0 then calling strncat has no effect.
• The function returns the value dest (i.e., a copy of the original dest pointer).
Restriction: you may not call any other library functions in or elsewhere. You should implement strncat, strncpy by processing the strings (character arrays) directly.
In both strncpy and strncat functions you should define the array/pointer inside the function and at the end you should return the array/pointer

Question 4) We would like to write a C program that opens a file named on the **command line** and copies that file to stdout replacing each single whitespace character in the file with a ".” (period) character. The program should not add or remove any newline characters – those should appear in the output file exactly as they were in the input. However, every other whitespace character (including each individual blank or tab) should be replaced by a ".”. For this problem, a whitespace character is any character c for which the library function
isspace(c) returns true (1). **(35 points)**(C programming)
The program should accept the name of the file as a command line argument to the program.
You are not allowed to read the name of the file in the execution of the program.
Example: Suppose program argument (argv[1]) is sample.txt and the input file sample.txt contains
To be or not 2 be? Is that a question?
then the program should copy this transformed version of sample.txt to stdout: **To.be.or.not..2.be? Is.that.a.question?**
Answer this question in the following two parts.

a) Complete the following function so that it replaces all of the whitespace characters in the string (char array) s with a '.' character. The number of elements (characters) in the array s is given by the parameter len. Assume that len has an appropriate value (i.e., s has at least that many elements). Also assume that any needed header files are already #included in the code.

```
/* Replace each whitespace char in s[0] through s[len-1] with a '.'
(period) */
void replace_whitespace(char *s, int len) {}
```

b) Give an implementation of the program so that it opens the file whose name is given on the command line (argv[1]) and writes it to stdout with whitespace characters replaced by periods. If the program does not have exactly one argument on the command line (either the file name is missing or there are extra arguments), or if the file cannot be opened, the program should write an appropriate message to either stdout or stderr and terminate with an exit code of 1. If the program terminates successfully, it should have an exit code of 0. The program should read one line at a time, and you may assume that no input line has more than 100 characters, including the newline '\n' and the '\0' byte at the end. Use the replace_whitespace(s, len) function defined in part (a) to replace whitespace characters with '.'. You may define additional functions if you need them, but you are not required to do so. Assume that all needed header files have already been #included.
Hint: be sure not to clobber existing '\n' characters by replacing them with '.' and don't add extra newlines to the output. Remember that the replace_whitespace parameter len can have whatever value makes sense for what you are trying to do.

```
int main(int argc, char** argv) {
}
```

==========

Reference Information

Some of this information might be useful while answering questions on the exam.

Shell: Some of the tests that can appear in a [ ] or [[ ]] test command in a bash script:

- string comparisons: =, !=
- numeric comparisons: -eq, -ne, -gt, -ge, -lt, -le
- Shell variables: $# (number of arguments), $? (last command result), $@, $* (all arguments), $0, $1, ... (specific arguments)