Assignment 3

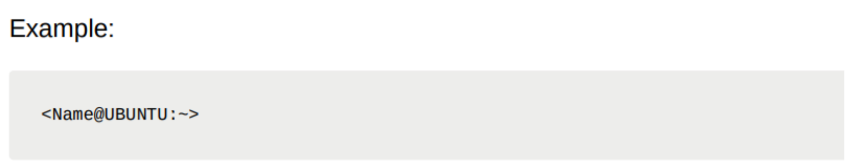
Deadline : 20th October 2024

The objective of this assignment is to develop a custom interactive. shell program. Please be aware that plagiarism will be closely monitored for this assignment, so do not copy code from others!

Specification 1: Display requirement [15 marks]

When you execute your code, a shell prompt of the following form must appear along with it. Do not hard-code the username and the system name here.

`Format : <username@system_name:curr_dir>`


Example:

```
<Name@UBUNTU:~>
```

The directory from which the shell is invoked will be the home directory of the shell and should be indicated by ~. If the user executes `cd i.e changes the directory`, then the corresponding change must be reflected in the shell as well.

Builtin commands [40 marks]

In your assignment, you're working with special commands that are part of the shell itself, like ‘pwd’ ,'echo', 'cd' and 'history'. When you use these commands in your shell, they run directly without needing external programs.

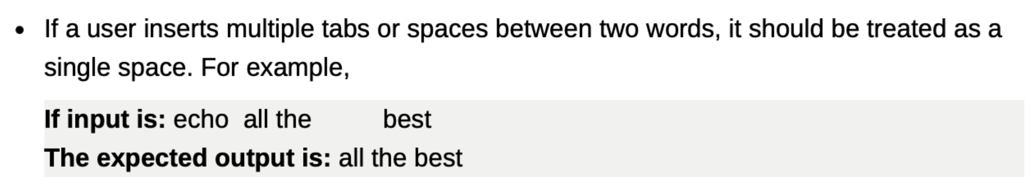
You have to make these three commands work in your shell. But, here's the catch:

you
can't use certain shortcut methods like `execvp` to make them work. You need to write the code for these commands within your shell program.

ECHO (10 MARKS):

The echo command in Linux is a built-in command that allows users to display lines of text
or strings passed as arguments.

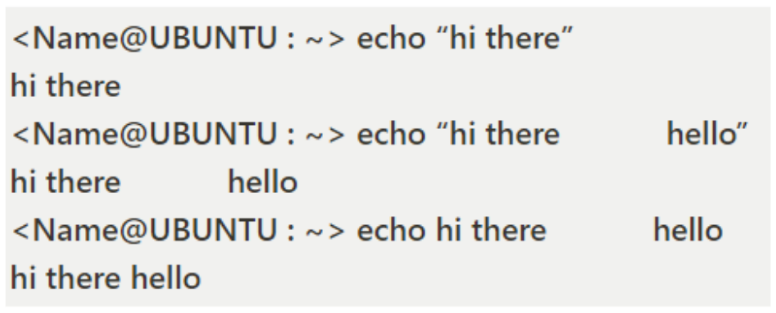
- For `echo`, handling multi-line strings and environmental variables is not a requirement.
- You do not need to handle escape flags and quotes.
- You can print the string as it is. However, you must handle tabs and spaces.
- If a user inserts multiple tabs or spaces between two words, it should be treated as a
  single space. For example,



- If a user inserts multiple tabs or spaces between two words, it should be treated as a single space. For example,

If input is: echo all the   best
The expected output is: all the best

You can test this on Ubuntu's actual terminal to see the expected output



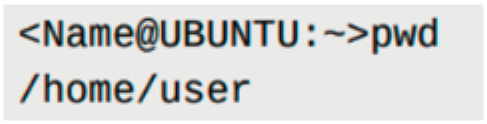
```
<Name@UBUNTU : ~> echo “hi there”
hi there
<Name@UBUNTU : ~> echo “hi there          hello”
hi there          hello
<Name@UBUNTU : ~> echo hi there          hello
hi there hello
```

Reference: https://linux.die.net/man/1/echo

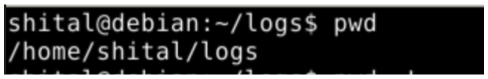
PWD: (10 MARKS)

PWD stands for Print Working Directory. It prints the path of the working directory, starting from the root.

The pwd command writes the full pathname of the current working directory to the standard output. Basically, the absolute path of the directory.



```
<Name@UBUNTU:~>pwd
/home/user
```



```
shital@debian:~/logs$ pwd
/home/shital/logs
```

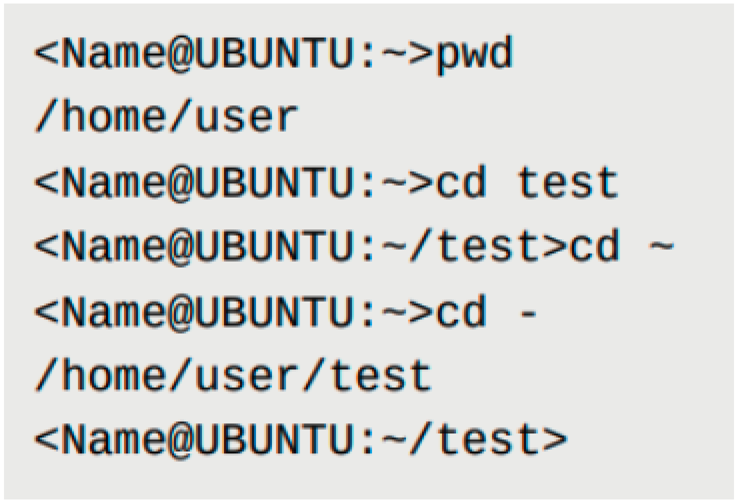
Reference : https://man7.org/linux/man-pages/man1/pwd.1.html

CD : (20 MARKS)

You'll need to implement the cd command which stands for `change directory`. This command allows the user to change the current working directory within your shell. Here
are some key details to consider:

- **cd :** We do not provide any arguments to the cd command to go to the home directory.
- **cd . . :** To shift one level above the current directory, we input .. as the argument.
- **cd - :** To go to the previous directory, we use `-` as our argument. This would first print the previous path and then go to past directory(refer example) .
- **cd ~ :** This takes the user to the home directory.

Error Handling: It's important to handle errors properly. If the cd command has more
than one argument, it should be considered an error.



```
<Name@UBUNTU:~>pwd
/home/user
<Name@UBUNTU:~>cd test
<Name@UBUNTU:~/test>cd ~
<Name@UBUNTU:~>cd -
/home/user/test
<Name@UBUNTU:~/test>
```

Reference :

https://manpages.ubuntu.com/manpages/trusty/man1/cd.1posix.html

BONUS (15 MARKS) :

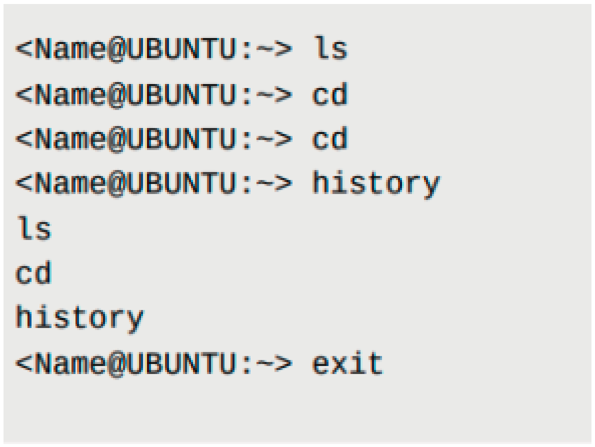
1: You can earn up to 5 bonus marks for creating a Makefile and ensuring modular, well-structured code.

Reference: https://makefiletutorial.com/

2: HISTORY (10 Marks)

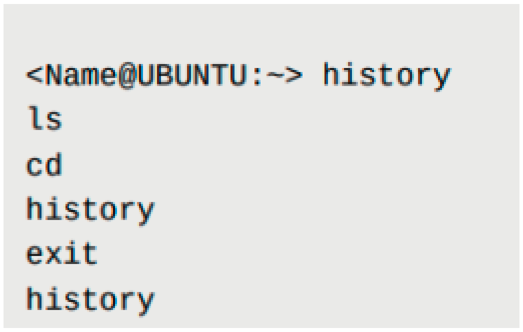
- Implement a `history command` which is similar to the actual history command.
- The default number of commands it should output is 10. The default number of commands the history should store is 20.
- You must overwrite the oldest commands if more than the set number of commands are entered.
- You should track the commands across all sessions and not just one.

DO NOT store the command in history if it is the exactly same as the previously entered command.



```
<Name@UBUNTU:~> ls
<Name@UBUNTU:~> cd
<Name@UBUNTU:~> cd
<Name@UBUNTU:~> history
ls
cd
history
<Name@UBUNTU:~> exit
```

WHEN YOU RUN SHELL AGAIN:



```
<Name@UBUNTU:~> history
ls
cd
history
exit
history
```

Reference : https://www.man7.org/linux/man-pages/man3/history.3.html

GUIDELINES:

- The Assignment must ONLY be done in C. NO other languages are allowed.
- All C standard library functions are allowed unless explicitly banned. Third-party
  libraries are not allowed.

- If the command cannot be run or returns an error it should be handled appropriately.
  Look at “perror.h” for appropriate routines to handle errors.
- You can use both “printf” and “scanf” for this assignment.
- If the code doesn't compile, no marks will be rewarded.
- Segmentation faults at the time of grading will be penalized.
- Write this code in a modular fashion (BONUS 5 MARKS AS STATED ABOVE)

SUBMISSION FORMAT :

1. Upload format <Roll-No>_Assignment3.tar.gz

2. Kindly adhere to the following naming guidelines and directory structure:

```
<Roll-No>_Assignment3
├──
README.md
└── makefile
└── Other files and Directories
```