System Programming

1. General Description of Lab:

Implement a simple user shell for UNIX that provides command execution, I/O redirection, pipe, and environment handling. The shell is a much simplified version of the standard Unix Bourne Again Shell, bash.

2. Constraints for this lab

This lab should be done individually; any copy from other students is forbidden and will receive zero point from this lab. The name of the shell will be name ush, I will give the basic code for you, and you are encouraged to implement your own version or base your code on our ICS lab5 code, as long as your shell satisfies my need and my defined interface standard.

3. Detailed requirement specifications

All ush features are listed below. If questions remain after reading the descriptions there, then ask.
Usage: ush [flags] [-c command]

Simple command	See sh man page for definitions
I/O redirection	<, >, and >>, with or without file descriptors
Command name validity checking	Verify that the command can be invoked Verify syntax for all implemented internal commands
Environment processing	= to assign, set, unset, export
Variable substitution	$name only
Pipe	| should connect two process together as a pipe

The following internal commands are to be supported.

- assignment, =
- cd
- echo
- exit
- export
- pwd
- set
- unset
- pipe, |

The following variables are defined on entry to the shell

- HOME - from calling environment
- PATH - from calling environment

3.1 Notes

- The commands **echo** and **pwd** work in the base code because both are implemented as external commands in /bin. For credit, they must be impelmented as internal commands.
- Variable substitution in identifiers (word token) only; not in quoted strings.
- The execvp is a very useful function. If the command is not built-in, execute the command using execvp and check the returned value from execvp(). If there is an error, print an appropriate error message based on errno.
- You should submit a short but clear document to tell me your design and implementation consideration.
- You need not to handle any case related to the pipe, which will be mentioned in chapter 14.
- Be careful that you wait for all processes in a pipe to complete before going on (that is, unless you have specified with & that you don't want to wait). Just to reiterate, you can have any number of processes piped together. For example, the following should work (it will reverse the file f1 and put the result in f2):

```
ush: cat -n f1 | sort -nr | sed s/...../// | cat > f2
```

3.2 Files to be modified

Your main work is to implement some function already declared in ush.c and ush-env.c, and you can modify other functions or add some functions on the condition that your should not change the interface of the shell and you must describe your strategy in your documents.