Exercise

- **Make simple mini shell**
  - This program should be under an infinite loop with conditional exit (“exit”)
    - » If a number is passed to exit, it should exit with that status
    - » If no number is passed, 0 is used as the exit status
    - » It is guaranteed that the argument is a number
  - When a command is entered, the command is executed using the child process
  - When the program quits, the parent process must wait for all child processes to terminate before exiting itself
  - The mini shell only executes programs under $PATH or the current directory
  - There is no limit to the header file, but you have to use exec* functions (i.e. system() cannot be used, commands should not be passed to shell in execvp, etc).
  - If the executable does not exist in $PATH or the current directory,
    - » print “executable name: command not found”

Exercise

```
$ ls -l
total 40
-rw-rw-r-- 1 spl spl   345 Apr  1 11:43 Makefile
-rwxrwxr-x 1 spl spl 17040 Apr  1 11:43 w5
-rw-rw-r-- 1 spl spl  4373 Apr  1 11:43 week5.c
-rw-rw-r-- 1 spl spl  6440 Apr  1 11:43 week5.o
$ head -n 3 week5.c
#include <linux/limits.h>       // PATH_MAX
#include <readline/history.h>   // add_history
#include <readline/readline.h>  // readline
$ bc -q
1 + 1
2
quit
$ make clean
rm -f week5.o w5
$ ls
Makefile  week5.c
$ make -s
$ something
something: command not found
$ ls -l
total 40
-rw-rw-r-- 1 spl spl   345 Apr  1 11:43 Makefile
-rwxrwxr-x 1 spl spl 17040 Apr  1 11:45 w5
-rw-rw-r-- 1 spl spl  4373 Apr  1 11:43 week5.c
-rw-rw-r-- 1 spl spl  6440 Apr  1 11:45 week5.o
$ bash -c ./w5
$ exit 1
exit
$ exit
exit
$ exit
exit
```

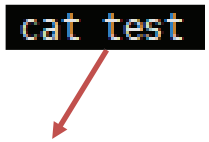
Exercise hint

- **readline and strtok_r functions**

```c
#include <linux/limits.h> // PATH_MAX
#include <readline/history.h>   // add_history
#include <readline/readline.h>  // readline
#include <stdio.h> // printf
#include <stdlib.h>  // free
#include <string.h>  // strtok_r
#define MAX_ARGS 1000

int main() {
  // readline
  char* cmd = readline("$ ");

  if (cmd == NULL) {
    printf("Error: Failed to read input\n");
    exit(1);
  }
  add_history(cmd);
  printf("You entered: %s\n", cmd);
  . . .
```



Exercise hint

```
  // strtok_r
  char* args[MAX_ARGS];
  char* save_ptr;
  char* ptr = strtok_r(cmd, " ", &save_ptr);
  int i = 0;
  while (ptr != NULL) {
    if (i >= MAX_ARGS - 1) {
      printf("Error: Too many arguments\n");
      exit(2);
    }
    args[i++] = ptr;
    ptr = strtok_r(NULL, " ", &save_ptr);
  }
  args[i] = NULL;

  for (int j = 0; j < i; j++) {
    printf("args[%d] = %s\n", j, args[j]);
  }
  // You have to free the memory allocated by readline
  free(cmd);
 return 0;
}
```

Exercise hint

- **When using readline,**
  - You have to add `-lreadline` when linking in the Makefile
    - » i.e. `gcc (obj file that uses readline).o –lreadline`
- **A shell operates using REPL (read-eval-print loop)**
  - **R**ead: get the input from user
  - **E**val: evaluate the input
  - **P**rint: print the result
  - **L**oop: Go back to the first step
- **When testing your code, try using programs that take user input (bc, man, python3, etc.)**

Exercise submission

- **Submit your source code and Makefile**
  - via iCampus
  - Bundle *source code* and *Makefile* with tar command assuming you are in the parent of the week5 directory.
    - » *tar.gz* format

    $ tar cvzf *STUDENT_ID.tar.gz week5*
  - We'll grade your submission with **make**
    - » If compilation fails, your points for this exercise will be zero
  - `vim TAR_FILE` should look similar

```
" tar.vim version v32a
" Browsing tarfile /home/spl/2024000000_홍길동.tar.gz
" Select a file with cursor and press ENTER

week5/
week5/week5.c
week5/Makefile
```