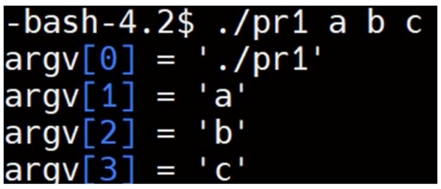
7376 HW1

Name: Xuebao Zhao   NUID: 002108354

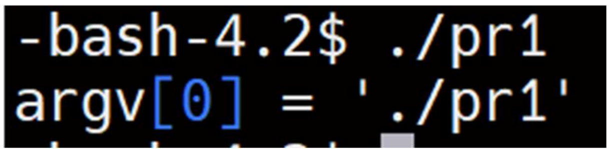
Problem1:

Function "print_args" takes the exact same arguments as function main. Then it uses a loop to traverse the array "argv" and print them one by one.

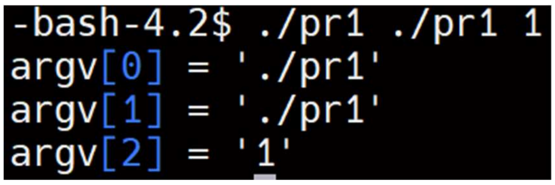
Results:



```
-bash-4.2$ ./pr1 a b c
argv[0] = './pr1'
argv[1] = 'a'
argv[2] = 'b'
argv[3] = 'c'
```



```
-bash-4.2$ ./pr1
argv[0] = './pr1'
```

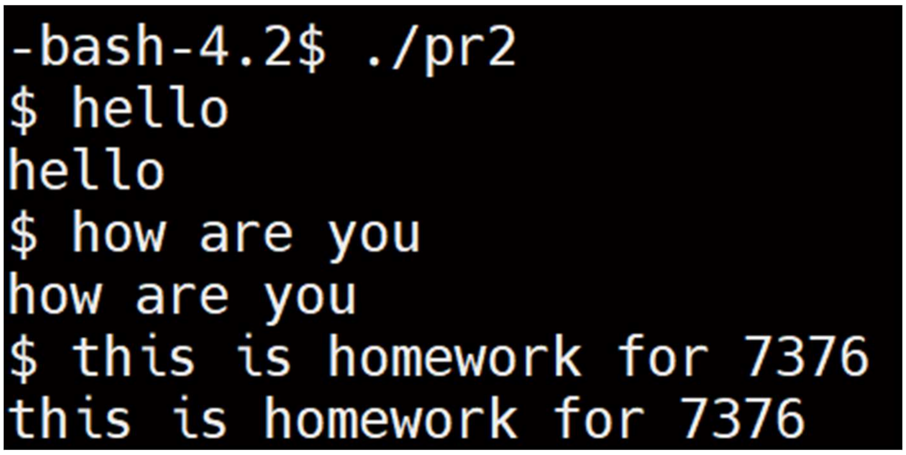


```
-bash-4.2$ ./pr1 ./pr1 1
argv[0] = './pr1'
argv[1] = './pr1'
argv[2] = '1'
```

Problem2:

This program uses function "printf" to print the prompt "$". Then it uses the function "fgets" to get the input from keyboard and store the input string into variable "ret". When the ret is not empty, the program gets the length of the string by using function "strlen" and replace the character('\n') with a null character ('\0'). Finally put the above statements into a "while (1)".

Results:



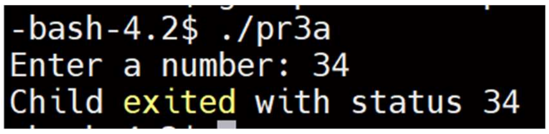
```
-bash-4.2$ ./pr2
$ hello
hello
$ how are you
how are you
$ this is homework for 7376
this is homework for 7376
```

Problem3:

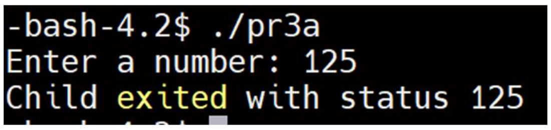
a)

At the beginning of the program, it creates the child process by using fork(). If folk failed, exit the program. When this program continues run, there is one parent process and one child process active in the system. The child process reads an integer value by using function "scanf" from the keyboard and returns it with an invocation of function "exit" (exit(input)). The parent process waits for the child process to finish and captures the child's exit status (wait(&status);). Finally, parent print the equal value.

Results:



```
-bash-4.2$ ./pr3a
Enter a number: 34
Child exited with status 34
```

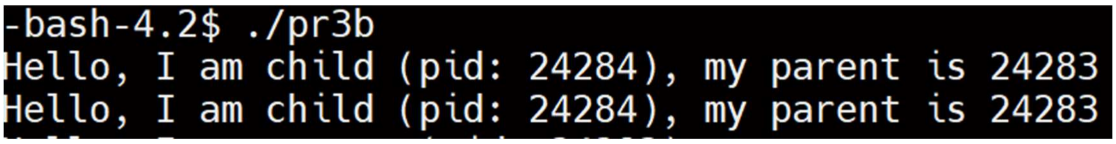


```
-bash-4.2$ ./pr3a
Enter a number: 125
Child exited with status 125
```

b)

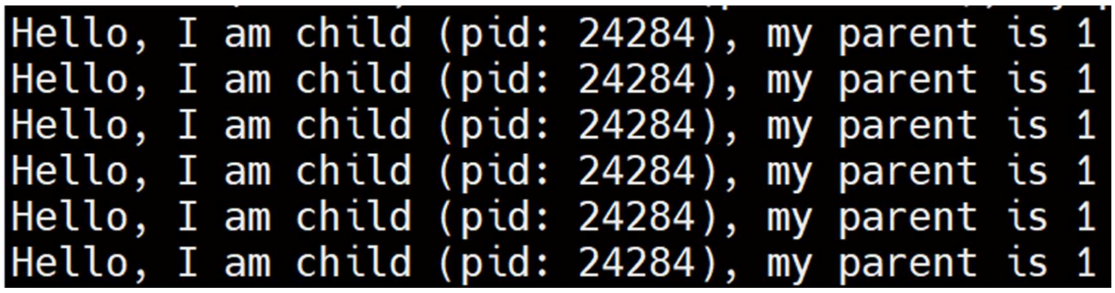
Change the code in the child process. Print the pid of the child process and the parent, and repeat the above statements after sleeping for two seconds (put them in a while(1) loop). The parent process will print its own information after sleeping for 4 seconds and end.

Results:



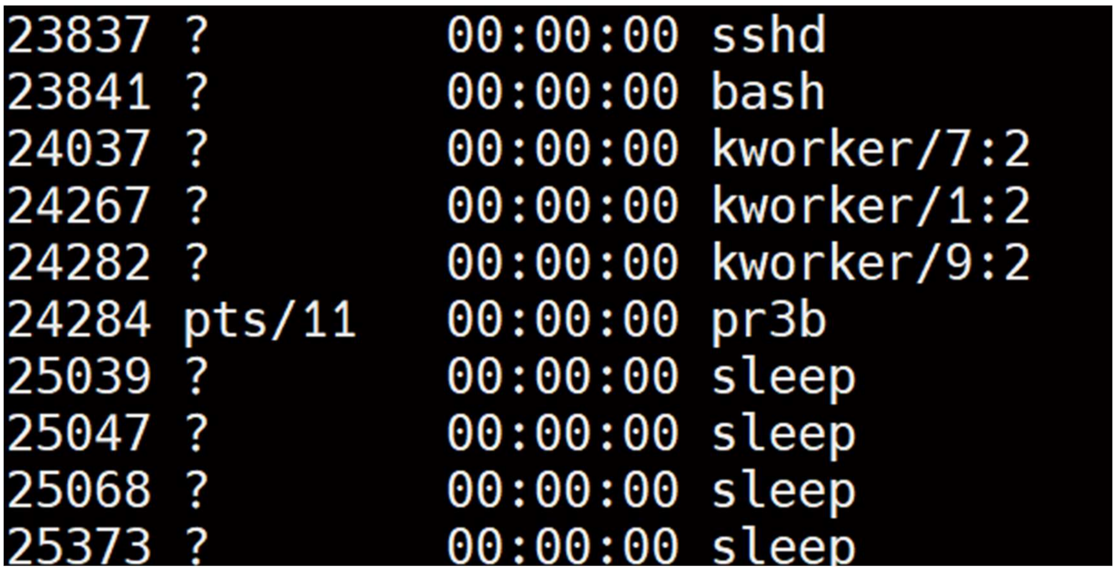
```
-bash-4.2$ ./pr3b
Hello, I am child (pid: 24284), my parent is 24283
Hello, I am child (pid: 24284), my parent is 24283
```

At first, the parent process and the child process are both running.



```
Hello, I am child (pid: 24284), my parent is 1
Hello, I am child (pid: 24284), my parent is 1
Hello, I am child (pid: 24284), my parent is 1
Hello, I am child (pid: 24284), my parent is 1
Hello, I am child (pid: 24284), my parent is 1
Hello, I am child (pid: 24284), my parent is 1
```

The parent process ends after 4 seconds, and the child process becomes an orphan process.



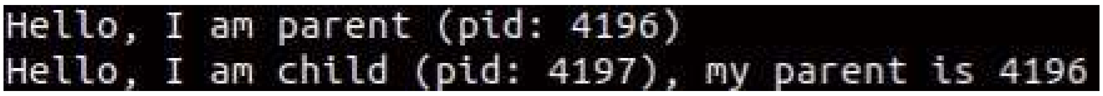
```
23837 ?        00:00:00 sshd
23841 ?        00:00:00 bash
24037 ?        00:00:00 kworker/7:2
24267 ?        00:00:00 kworker/1:2
24282 ?        00:00:00 kworker/9:2
24284 pts/11   00:00:00 pr3b
25039 ?        00:00:00 sleep
25047 ?        00:00:00 sleep
25068 ?        00:00:00 sleep
25373 ?        00:00:00 sleep
```

The child process (pid = 24284, name = pr3b) still exist, but the parent process cannot be found.

c)

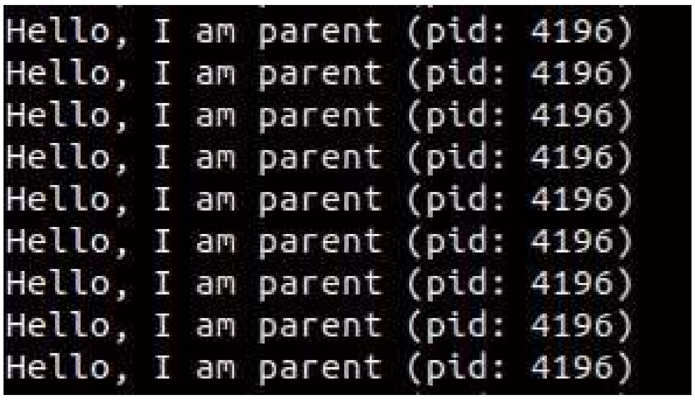
Do the same thing as pr3b, but swap what is done in the child process with the parent process. The child process print the pid of the child process and the parent only once. The parent process will print its own information after sleeping for 2 seconds and repeat (put them in a while (1) loop).

Results:



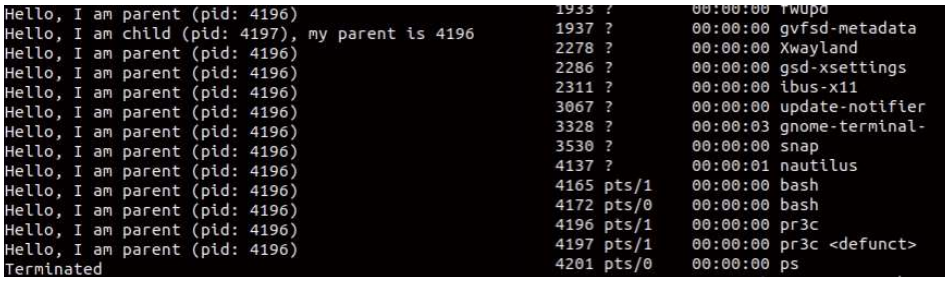
```
Hello, I am parent (pid: 4196)
Hello, I am child (pid: 4197), my parent is 4196
```

At first, the parent process and the child process are both running.



```
Hello, I am parent (pid: 4196)
Hello, I am parent (pid: 4196)
Hello, I am parent (pid: 4196)
Hello, I am parent (pid: 4196)
Hello, I am parent (pid: 4196)
Hello, I am parent (pid: 4196)
Hello, I am parent (pid: 4196)
Hello, I am parent (pid: 4196)
Hello, I am parent (pid: 4196)
```

The child process ends.



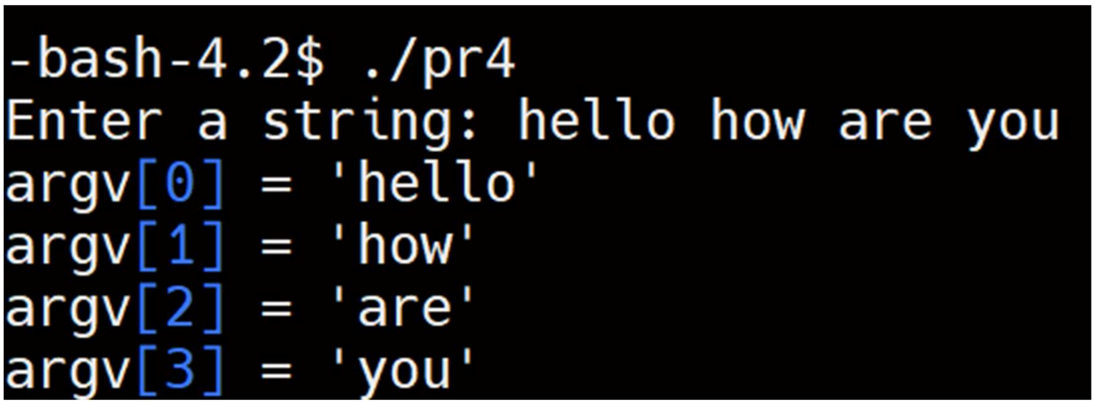
```
Hello, I am parent (pid: 4196)                          1933 ?        00:00:00 fwupd
Hello, I am child (pid: 4197), my parent is 4196        1937 ?        00:00:00 gvfsd-metadata
Hello, I am parent (pid: 4196)                          2278 ?        00:00:00 Xwayland
Hello, I am parent (pid: 4196)                          2286 ?        00:00:00 gsd-xsettings
Hello, I am parent (pid: 4196)                          2311 ?        00:00:00 ibus-x11
Hello, I am parent (pid: 4196)                          3067 ?        00:00:00 update-notifier
Hello, I am parent (pid: 4196)                          3328 ?        00:00:03 gnome-terminal-
Hello, I am parent (pid: 4196)                          3530 ?        00:00:00 snap
Hello, I am parent (pid: 4196)                          4137 ?        00:00:01 nautilus
Hello, I am parent (pid: 4196)                          4165 pts/1    00:00:00 bash
Hello, I am parent (pid: 4196)                          4172 pts/0    00:00:00 bash
Hello, I am parent (pid: 4196)                          4196 pts/1    00:00:00 pr3c
Hello, I am parent (pid: 4196)                          4197 pts/1    00:00:00 pr3c <defunct>
Terminated                                              4201 pts/0    00:00:00 ps
```

The child process is a zombie and indicated by <defunct>

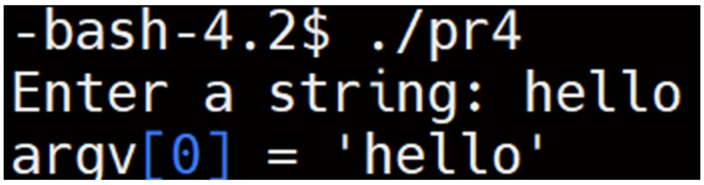
Problem4:

Use the same method as problem2 to read a string from the user. Then extract arguments in function “get_args”. The function “get_args” defines a string “temp” to temporarily store the fragment of argument. Every time using function “strtok” to get an fragment and store it in temp. When temp is not empty, copy it to “argv” and the variable “num” which counts the number of fragments plus one. Continue until temp equals to empty.

Results:



```
-bash-4.2$ ./pr4
Enter a string: hello how are you
argv[0] = 'hello'
argv[1] = 'how'
argv[2] = 'are'
argv[3] = 'you'
```



```
-bash-4.2$ ./pr4
Enter a string: hello
argv[0] = 'hello'
```