Quiz 1

Operating Systems, Fall 2019
Iran University of Science and Technology

Solutions

Problem 0: How many '*'s are printed on the screen. Assume all *fork* system calls succeed.

part 1)

```c
#include <stdio.h>
#include <unistd.h>


int main(int argc, char *argv[])
{
  printf("*\n");
  fflush(stdout);
  int rc;
  for (int i = 0; i < 3; i++) {
    rc = fork();
    printf("*\n");
    fflush(stdout);
    if (rc > 0)
      wait(rc);
  }
  return 0;
}
```

p1.c

Note: *fflush* is used to make sure the write buffer is cleaned before forking.

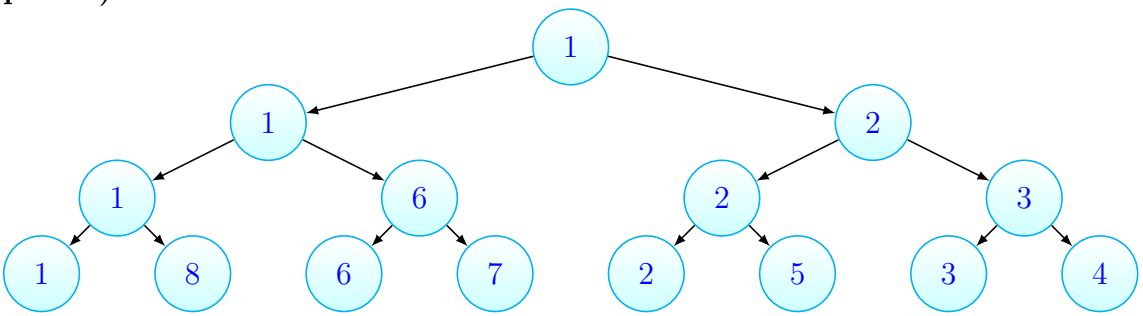
part 2)

```c
#include <stdio.h>
#include <unistd.h>


int main(int argc, char *argv[])
{
   fork();
   fork();
   fork();
   printf("*\n");
   return 0;
}
```

p2.c

Answer 0:
part 1)



In the above graph, each node represents a process. Each node is numbered with its process-id. At first there exists one process executing the main function. This process is shown as the root of the tree. Then when the first *fork* is called there will be two processes. each process will now execute print statement and after that they will call *fork* again. In the end, there will be 8 process. Both parent and child process will execute the print statement. The parent process has executed print once before the loop.
So the number of printed '*'s are equal to the number of nodes. There will be 15 '*'s on the screen.

part 2)

After executing the code in part 2, there will be eight processes. Same as part 1, each child process will fork new processes. In this part each process will print a star, as a result, there will be eight '*'s on the screen.

Problem 1: When writing a function in bash how can we access the passed arguments.

Answer 1:
Arguments passed to a function can be accessed with experation '$n'. Where 'n' is the number of the argument to be accessed. 'n' starts from one.
The total number of arguments passed to the function is stored in '$#'.

```
function test() {
        echo "First arg = $1"
        echo "Second arg = $2"
        echo $#
}
```

Problem 2: Write a for loop in bash to print the name of files in the current directory.

Answer 2:

```
for f in $( ls )
do
        echo $f
done
```

For walking through the subdirectories:

```
function walk() {
        for f in $( ls $1)
        do
                ADD="$1/$f"
                echo $ADD
                # check if Address is a directory
                if [ -d $ADD  ]
                then
                        walk $ADD
                fi
        done
}

walk .
```