Operating System Lab
Assignment 3
Arnav Samal
122CS0107

Q1.
Code:

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main() {
    fork();
    fork();
    printf("Hierarchy of 3 Processes\n");
    return 0;
}
```

Output:

```
r-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OperatingSystem/Lab_3$ gcc q1.c
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OperatingSystem/Lab_3$ ./a.out
Hierarchy of 3 Processes
Hierarchy of 3 Processes
Hierarchy of 3 Processes
Hierarchy of 3 Processes
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OperatingSystem/Lab_3$
```

Q2.

Code:

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

#define MAX_PROCESSES 7

int processTree[MAX_PROCESSES];
int pid;

void printTree(int index, int level) {
    if (index >= MAX_PROCESSES) return;
    printTree(2*index+2, level+1);
    for (int i = 0; i < level; i++) printf("  ");
    printf("Process %d\n", processTree[index]);
    printTree(2*index+1, level+1);
}

int main() {
    printf("Here I am just before first forking statement\n");
    pid = fork();
    processTree[0] = getpid();
    if (pid == 0) {
        printf("Here I am just after first forking statement\n");
        pid = fork();
        processTree[1] = getpid();
        if (pid == 0) {
            printf("Here I am just after second forking statement\n");
            pid = fork();
            processTree[3] = getpid();
            if (pid != 0) {
                processTree[4] = getpid();
            }
        } else {
            processTree[2] = getpid();
            pid = fork();
            processTree[5] = getpid();
            if (pid != 0) {
                processTree[6] = getpid();
            }
        }
    }
    printf("Hello World from process %d!\n", getpid());
    printTree(0, 0);
    return 0;
}
```

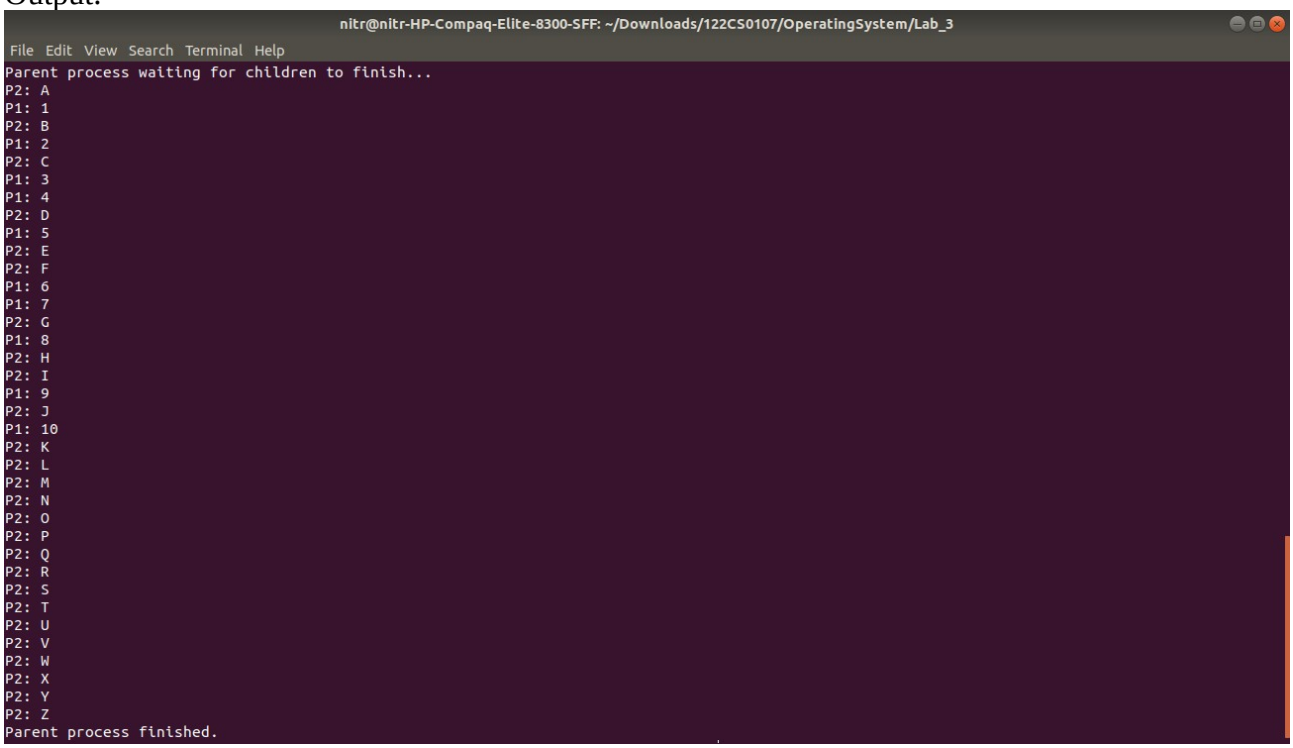
Output:

```
nitr@nitr-HP-Compaq-Elite-8300-SFF: ~/Downloads/122CS0107/OperatingSystem/Lab_3
File  Edit  View  Search  Terminal  Help
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OperatingSystem/Lab_3$ ./a.out
Here I am just before first forking statement
Hello World from process 2965!
    Process 0
  Process 0
    Process 0
Process 2965
    Process 0
  Process 0
    Process 0
Here I am just after first forking statement
Hello World from process 2966!
    Process 2966
  Process 2966
    Process 2966
Process 2966
    Process 0
Here I am just after second forking statement
  Process 2966
    Process 0
Hello World from process 2968!
    Process 0
  Process 2966
    Process 2968
Process 2966
    Process 0
  Process 2966
    Process 0
Hello World from process 2967!
    Process 0
  Process 0
    Process 0
Process 2966
    Process 2967
  Process 2967
    Process 2967
Hello World from process 2969!
```

Q3.
Code:

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main() {
    pid_t pid1, pid2;

    pid1 = fork();
    if (pid1 == 0) {
        for (int i = 1; i <= 10; i++) {
            printf("P1: %d\n", i);
            sleep(1);
        }
        return 0;
    }

    pid2 = fork();
    if (pid2 == 0) {
        for (char c = 'A'; c <= 'Z'; c++) {
            printf("P2: %c\n", c);
            sleep(1);
        }
        return 0;
    }

    printf("Parent process waiting for children to finish...\n");
    waitpid(pid1, NULL, 0);
    waitpid(pid2, NULL, 0);
    printf("Parent process finished.\n");

    return 0;
}
```

Output:

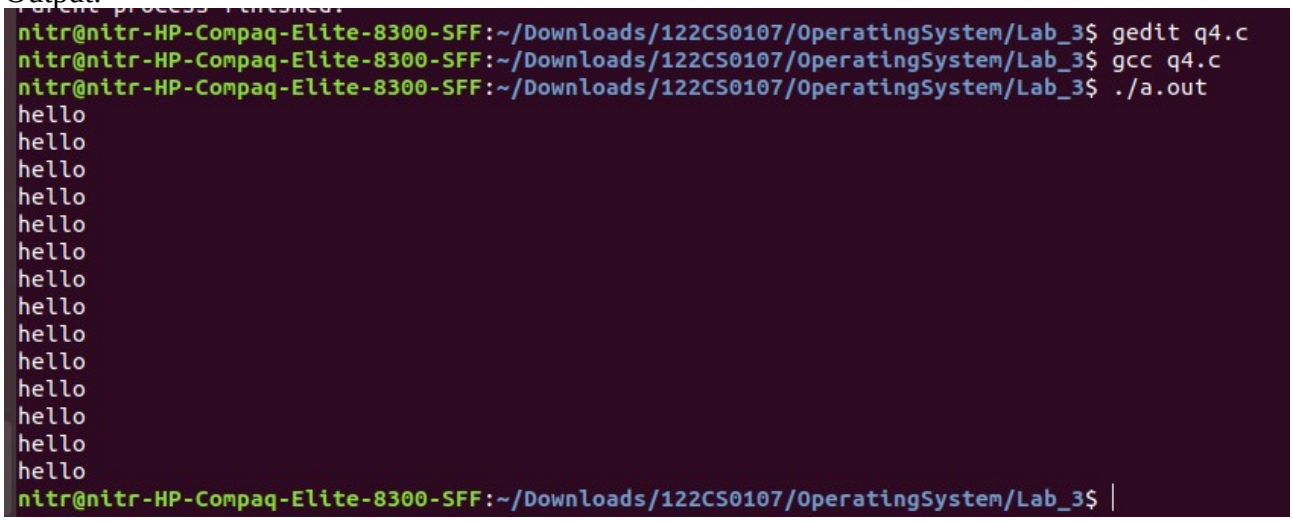


Q4.

Code(given):

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main(void)
{
        int i; for (i=0; i < 3; i++) {
        fork();
        printf("hello\n");
        }
        return 0;
}
```

Output:



Explanation:

When the code is initially run, it is first forked to make 2 processes child process and copy of parent process. These then print “hello”, and then it is forked again where 4 processes then print “hello”. This goes on one last time where it is forked for 3rd time and then 8 processes then print “hello”. In total 2+4+8 = 14 “hello”s are printed.