**Lab 6 Stacks and the Fork Command**

Q1 Using the notes and other sources write C code that implements a queue data structure for processes. You can assume that data element of the queue node is an integer process ID.

**Write a program with the following functions that**

**Enqueue a node** *(add a node to beginning of the queue)*

**Dequeue a node *(remove a node from the end of a queue)***

**A menu function** *that allows the user to choose if they want to:*

1. **Add a process to queue**
2. **Remove a process from the Queue**
3. **Exit the program**

**Q2** Run the sample code **fork2.c**. explain the reason for any difference.

**Q3:**

1. **Copy the code and run the following explain output.**
2. **Comment out the sleep command in the child process and rerun the program explain the output. :**

*#include <unistd.h>*

*#include <stdio.h>*

*int main()*

*{*

*int i;*

*printf("hello before fork \n");*

*printf("i : %d\n",i);*

*i=fork();*

*printf("\n");*

*if(i==0)*

*{*

*printf("Child has started\n\n");*

*printf("child printing first time \n");*

*printf("getpid : %d getppid : %d \n",getpid(),getppid());*

*sleep(5);*

*printf("\nchild printing second time \n");*

*printf("getpid : %d getppid : %d \n",getpid(),getppid());*

*}*

*else*

*{*

*printf("parent has started\n");*

*printf("getpid : %d getppid : %d \n",getpid(),getppid());*

*printf("\n");*

*}*

*printf("Hi after fork i : %d\n",i);*

*return 0;*

}

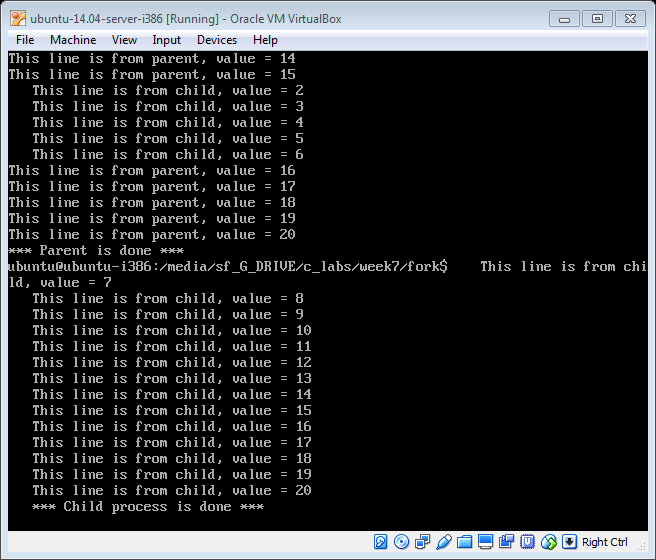
Run this program a number of times and explain, in detail, the output. Do not forget to comment the code to explain what each part is doing.

Q4 ----------------------------- Home Work (if not finished) --------------------------

Set up virtual box and the Ubuntu server on the desktop or on your laptop. Refer to lab 1 for instructions. Rerun Q2 and Q3 on the virtual box and observer the output.

1. **Is there a difference between the outputs?**
2. **Why is there a difference between them**

Q3 Sample output on virtual box



Q4 sample output on virtual box

