Roll No: 2003154

Lab Evaluation
Set - Z

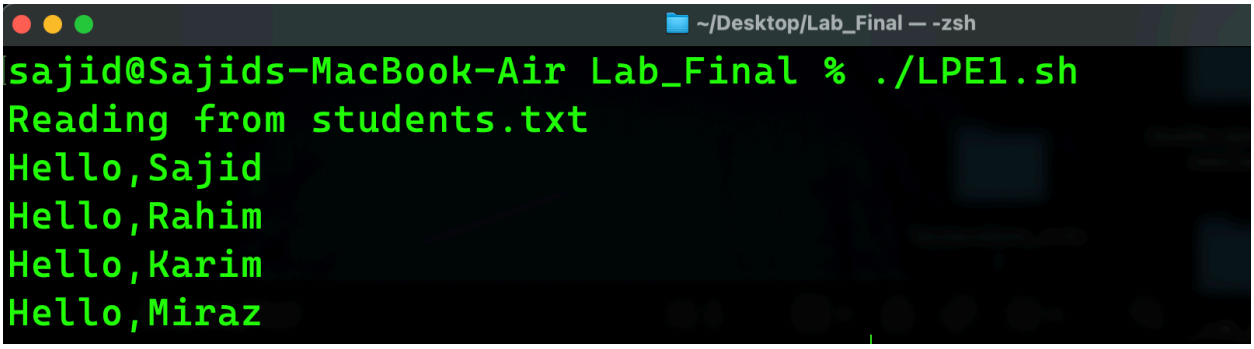
Lab Task Q[1]

Question: Take input from a file students.txt and print hello for each of the student.

Solution (Code):

```
#!/bin/bash
echo "Reading from students.txt"
while read line
do
    echo "Hello,"$line
done < students.txt
```

Output (Screenshot/SnapShot):



Lab Task Q[2]

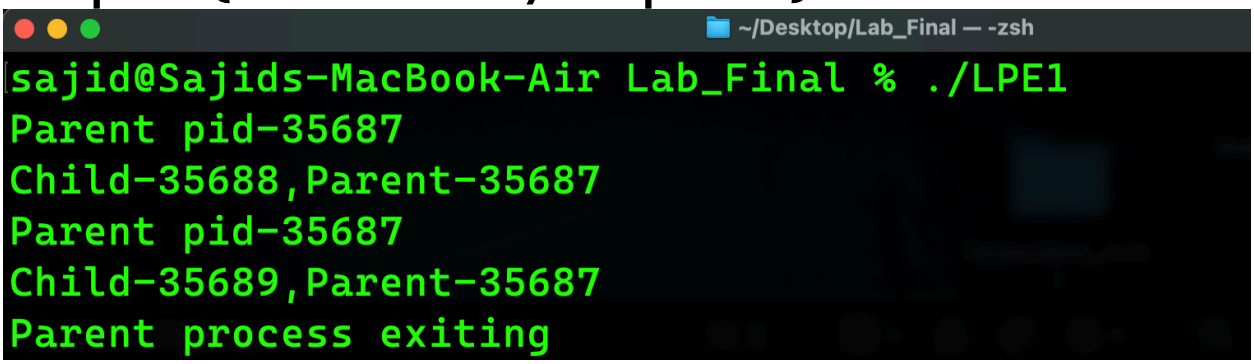
Question: Write a system program where a parent process creates exact two child processes and printing child pid and parent pid. Parent process exiting after all the child terminate.

Solution (Code):

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

const int numChildren = 2;

int main() {
    int j;
    pid_t childPid;
    setbuf(stdout, NULL);

    for (j = 0; j < numChildren; j++) {
        switch (childPid = fork()) {
        case -1:
```

```
            printf("Error: fork
failed\n");
            return EXIT_FAILURE;
        case 0:
            printf("Child-%d,Parent-
%d\n", getpid(),getppid());
            exit(EXIT_SUCCESS);
        default:
            printf("Parent pid-%d\n",
getpid());
            wait(NULL); //waiting for the
child process to finish
            break;
        }
    }
    printf("Parent process exiting\n");

    return EXIT_SUCCESS;
}
```

Output (Screenshot/SnapShot):



Lab Final
Lab Task Q[3]

Question: Create two processes named producer1, producer2 sharing the same global variable "i" which is initialized by 0. Producer1 increments i by one and producer2 increments i by two. Producer2 is created first. Both processes terminates when i is exactly 6.

Solution (Code):

```
#include <xinu.h>

int i = 0;

void producer1(sid32 prod1, sid32 prod2)
{
    while (i <= 6)
    {
```

```
        wait(prod2);
        if (i > 6)
        {
            break;
        }
        printf("producer1:%d\n", i++);
        signal(prod1);
    }
}
void producer2(sid32 prod1, sid32 prod2)
{
    while (i <= 6)
    {
        wait(prod1);
        printf("producer2:%d\n", i);
        i = i + 2;
        signal(prod2);
    }
}
int hello()
{
    sid32 prod1 = semcreate(1);
    sid32 prod2 = semcreate(0);
```

```
    resume(create(producer1, 1024, 20,
"prod1", 2, prod1, prod2));
    resume(create(producer2, 1024, 20,
"prod2", 2, prod1, prod2));

    return 0;
}
```

Output (Screenshot/SnapShot):