Pratham Sharma, AC-1232
Operating System, Semester 3
Assignment 4 - Forking

Program 1 :

```cpp
#include <iostream>
#include <stdio.h>
#include <unistd.h>
using namespace std;

int main() {
    printf( "Following line of code will run 2^3 times \n\n");
    /* fork a child process */
    fork();
    fork();
    fork();

    cout << "Fork" << endl;

    return 0;
}
```

Output :

```
┌──(root💀prathm)-[/mnt/c/Users/lenovo/code/ug/sem3/os/assignment4fork]
└─# g++ fork1.cpp && ./a.out
Following line of code will run 2^3 times

Fork
Fork
Fork
Fork

Fork
┌──(root💀prathm)-[/mnt/c/Users/lenovo/code/ug/sem3/os/assignment4fork]
└─# Fork
Fork
Fork


┌──(root💀prathm)-[/mnt/c/Users/lenovo/code/ug/sem3/os/assignment4fork]
└─#
```

Program 2 :

```cpp
#include <stdio.h>
#include <unistd.h>
using namespace std;

int main() {
    int n;
    printf("enter no of times you want to fork : ");
    scanf("%d",&n);
    printf("Following line will output 2^n times\n\n");

    for (int i = 0; i < n; i++){
        fork();
    }

    printf("hello\n");

    return 0;
}
```

Output :

```
┌──(root💀prathm)-[/mnt/c/Users/lenovo/code/ug/sem3/os/assignment4fork]
└─# g++ fork2.cpp && ./a.out
enter no of times you want to fork : 2
Following line will output 2^n times

hello
hello
hello
hello

┌──(root💀prathm)-[/mnt/c/Users/lenovo/code/ug/sem3/os/assignment4fork]
└─#
```

Program 3 :

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

int main() {
    pid_t pid;
    /* fork a child process */
    pid = fork();

    if (pid < 0) {/* error occurred */
        fprintf(stderr, "Fork Failed");
        return 1;
    }

    else if (pid == 0) {/* child process */
        execlp("/bin/ls","ls",NULL);
        printf("LINE J");
    }

    else {/* parent process */
        /* parent will wait for the child to complete */
        // wait(NULL);
        printf("Child Complete");
    }

    return 0;
}
```

Output :

```
┌──(root💀prathm)-[/mnt/c/Users/lenovo/code/ug/sem3/os/assignment4fork]
└─# g++ fork3.cpp && ./a.out
Child Complete
┌──(root💀prathm)-[/mnt/c/Users/lenovo/code/ug/sem3/os/assignment4fork]
└─# a.out  fork1.cpp  fork2.cpp  fork3.cpp      fork4.cpp  fork5.cpp
```

Program 4 :

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

int main() {
    pid_t pid, pid1;

    /* fork a child process */
    pid = fork();
    if (pid < 0) {/* error occurred */
        fprintf(stderr, "Fork Failed");
        return 1;
    }

    else if (pid == 0) {/* child process */
        pid1 = getpid();
        printf("\nchild: pid = %d",pid); /* A */
        printf("\nchild: pid1 = %d",pid1); /* B */
    }

    else {/* parent process */
        pid1 = getpid();
        printf("\nparent: pid = %d",pid); /* C */
        printf("\nparent: pid1 = %d",pid1); /* D */
        // wait(NULL);
    }
    return 0;
}
```

Output :

```
┌──(root💀prathm)-[/mnt/c/Users/lenovo/code/ug/sem3/os/assignment4fork]
└─# g++ fork4.cpp && ./a.out

parent: pid = 983

parent: pid1 = 982child: pid = 0
child: pid1 = 983
┌──(root💀prathm)-[/mnt/c/Users/lenovo/code/ug/sem3/os/assignment4fork]
└─#
```

Program 5 :

```c
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
using namespace std;

#define SIZE 5
int nums[SIZE] = {0,1,2,3,4};

int main() {
    int i;
    pid_t pid;
    pid = fork();

    if (pid == 0) {
        for (i = 0; i < SIZE; i++) {
            nums[i] *= -i;
            printf("\nCHILD: %d ",nums[i]); /* LINE X */
        }
    }
    else if (pid > 0) {
        // wait(NULL);
        for (i = 0; i < SIZE; i++)
            printf("\nPARENT: %d ",nums[i]); /* LINE Y */
    }
    return 0;
}
```

Output :

```
┌──(root💀prathm)-[/mnt/c/Users/lenovo/code/ug/sem3/os/assignment4fork]
└─# g++ fork5.cpp && ./a.out

PARENT: 0
PARENT: 1
PARENT: 2
PARENT: 3
PARENT: 4
CHILD: 0
CHILD: -1
CHILD: -4
CHILD: -9
CHILD: -16
┌──(root💀prathm)-[/mnt/c/Users/lenovo/code/ug/sem3/os/assignment4fork]
└─#
```