No E-Mail submissions will be accepted.
Submission formats and file naming:
File name : firstName_lastName_lab_4
File format: pdf or MS Word format
e.g. Donald_Trump_lab_4.pdf

Just for questions 1 and 2 use online Linux https://cocalc.com/

(Do not use your local machine)

1. First compile your code to obtain main1.out and then run main1.out using:

```
./main1.out & pstree -pT | grep main1.out
```

main1.c

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

int main(){

    pid_t fork_return = fork();
    pid_t  pid = getpid();
    printf("fork return value = %d , pid = %d \n", fork_return, pid);
    sleep(1);

    return 0;
}
```

Now answer the given questions below.

a) Attach a screenshot of your program's output after execution.

```
~$ ./main
fork return value = 992 , pid = 991
fork return value = 0 , pid = 992
```

b) Explain the purpose of the getpid() function. What does it return?
It returns the PID of the currently running process

c) Using the output from part (a), identify and write down the Parent Process ID (PID).
991

d) From the same output, identify and write down the Child Process ID (PID).
992

e) Based on the output, what is the return value of fork() in the parent process?

992

f) Based on the output, what is the return value of fork() in the child process?

0

2.Consider the following c code, first compile your code

```
gcc -static main2.c -o main2.out
```

to obtain the binary file main2.out and then run it.

main2.c

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

int global_variable = 100;
int un_global_variable;

int main()
{
    char str[30];
    int *ptr;
    int local_variable = 200;
    int un_local_variable;

    ptr = (int*) malloc(2 * sizeof(int));

    pid_t pid;
    pid = getpid();

    printf("\n");
    printf(" Global variable                : %p\n", &global_variable);
    printf(" Uninitialized global variable  : %p\n", &un_global_variable);

    printf(" Local variable                 : %p\n", &local_variable);
    printf(" Uninitialized local variable   : %p\n", &un_local_variable);

    printf(" Memory allocation              : %p\n\n", ptr);
    printf("  ======== The Process Address Space ( pid = %d )=======\n\n", pid);

    sprintf(str, "cat -b /proc/%d/maps", pid);
    system(str);
    free(ptr);

    return 0;
}
```

a) Attach a screenshot of your output.

```
~$ ./main2.out

Global variable                  : 0x4ac0d0
Uninitialized global variable    : 0x4adb30
Local variable                   : 0x7ffd87d7dbec
Uninitialized local variable     : 0x7ffd87d7dbf0
Memory allocation                : 0x1a34b900

======== The Process Address Space ( pid = 1506 )=======

    1  00400000-00401000 r--p 00000000 00:265 258                              /home/user/main2.out
    2  00401000-00481000 r-xp 00001000 00:265 258                              /home/user/main2.out
    3  00481000-004a7000 r--p 00081000 00:265 258                              /home/user/main2.out
    4  004a7000-004ac000 r--p 000a6000 00:265 258                              /home/user/main2.out
    5  004ac000-004ae000 rw-p 000ab000 00:265 258                              /home/user/main2.out
    6  004ae000-004b4000 rw-p 00000000 00:00 0
    7  1a34a000-1a36c000 rw-p 00000000 00:00 0                                 [heap]
    8  7ffd87d5e000-7ffd87d80000 rw-p 00000000 00:00 0                         [stack]
    9  7ffd87daa000-7ffd87dae000 r--p 00000000 00:00 0                         [vvar]
   10  7ffd87dae000-7ffd87db0000 r-xp 00000000 00:00 0                         [vdso]
   11  ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0                 [vsyscall]
```

b) Obtain the location of the initialized and uninitialized global variables (the line number)?
Line 5: 004ac000-004ae000

c) Obtain the location of the initialized and uninitialized local variables (the line number)?
Line 8: 7ffd87d5e000-7ffd87d80000

d) Obtain the location of the allocated memory (the line number)?
Line 7: 1a34a000-1a36c000

e.g.

line 34

```
33  7ffc7e75b000-7ffc7e75f000 r--p 00000000 00:00 0                  [vvar]
34  7ffc7e75f000-7ffc7e761000 r-xp 00000000 00:00 0                  [vdso]
35  ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0          [vsyscall]
```

3. A computer system can hold three identical processes in its main memory at a time.Each process has an independent probability p = 1/2.

A. Calculate the overall CPU utilization, defined as the probability that at least one process is ready to use the CPU.
1 - p^3 = 1 - (½)^3 = 87.5% one process will be waiting

B. Now consider only the scenario in which exactly one process is waiting for I/O, and the other two are ready. What is the CPU utilization in this specific case?

100% utilization since 2 processes are ready

4. Consider the following c code, compile your code to obtain the binary file main3.out.

```
gcc main3.c -o main3.out
```

main3.c

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

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

    if (pid < 0) {
        perror("fork failed");
        exit(1);
    }

    if (pid == 0) {
        printf("Process x \n");
        exit(0);
    } else {
        printf("Process y \n");
        sleep(30);
    }

    return 0;
}
```

Run your program and use its output to answer the following questions:

```
./main3.out & ps -e  -o pid,stat,comm | grep main3.out
```

A. Attach a screenshot of your program's output.

```
~$ ./main3.out & ps -e -o pid,stat,comm | grep main3.out
[1] 2074
Process y
Process x
   2074 SN   main3.out
   2077 ZN   main3.out
```

B. Identify which process (x/y) is the child and state its PID.
Process: x Pid: 2077

C. Identify which process (x/y) is the parent and state its PID.
Process: y Pid: 2074

D. Specify the process state of both the child and the parent.

Child: Defunct ("Zombie") Process, terminated but not reaped & Low Priority

Parent: Interruptible Sleep & Low Priority