QUESTION 1

Trace the following program and answer the following questions:

```
#include <pthread.h>
#include <stdio.h>
 …
pid_t pid;
 …
pid = fork();                    //Line A
if (pid == 0)
      pthread_create( . . .);     //Line B
else if (pid > 0) {
      Wait();
      fork();  }                  //line C
else
      pthread_create( . . .);     //line D
 …
}
```

- Assume that fork() always succeeds. After executing the fork() statement in line A, the total number of processes = 2 and the total threads= 2 .
- After executing line B, the total number of processes = 2 and the total number of threads= 3 .
- Assume that fork() always succeeds. After executing the fork() statement in line C, the total number of processes = 3 and the total number of threads= 4 .
- After executing line D, the total number of processes = 1 and the total number of threads= 2 .

QUESTION 2

Trace the following program and answer the following questions:

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

void *square (void *param);
int x = 10;

int main (int argc, int argv[]){
      pthread_t tid[3];

      for (int i=0; i<3; i++)
            pthread_create(&tid[i], NULL, *square, i+1);


      for (int i=0; i<3; i++)
            pthread_join (tid[i], NULL);
      }

void * square (void *param){
      int y=x*x ;
      printf ("The %d double is %d \n",param,y);
      pthread_exit(0); }
```

- How many threads are created by phread_create? 3
- Is the thread ID the same as the process ID (Yes/No)? No
- What is the name of the function that was executed by all threads? square
- What is the parameter that was passed to the thread function? i+1
- Answer with True or False:
  - Variable (x) is a Local shared variable among all threads. False
  - Variable (y) is a Local not shared variable among all threads. True
  - Variable (x) is a shared variable among all threads. True
  - The value of y of the first executed thread is 100. True
  - The value of y of the second executed thread is 10000. False
  - If the main program does not call thread_join() after creating a thread, then the main process will be blocked until all threads are terminated successfully. False

QUESTION 3

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

int main(void)
{
      pid_t pid = fork();
      wait(NULL);
     for (int i = 0;  i < 2;  i++)
            printf ("This is process %d\n", getpid());

      return 0;
}
```

Suppose the parent Id is 4444, child Id is 5555

- Give the output of this program (Ignore the \n in the output)? This is process 5555 This is proc

Insert **only** the **value in b and c** without adding any extra spaces or characters before or after

- What is the return value of the wait system call in the parent process? 5555
- What is the return value of the wait system call in the child process? -1

QUESTION 4

```
# include <stdio.h>
# include <stdlib.h>
# include <sys/types.h>
# include <unistd.h>
int main(int argc, char *argv[])
{
    pid_t pid;
    int m = 10;
    for (int i=0; i<3;i++){
       pid = fork();            // Line A
       if (pid ==0) {
            m*=i+1;
                  break;}
        if (pid == -1)
            exit(1);}
    if (pid == 0)
        printf ("%d ",m);        // Line B
    if (pid > 0) {
        wait(NULL);              // Line C
        printf ("Process 0 ");}
    return 0;
}
```

Note: Assume that Fork() always succeeds.

- How many times does fork()execute in LineA? 3
- How many processes were created in the program? 4
- How many times does the print statement in Line B will be executed? 3
- How many times does Line C execute? 1

Insertt **only** the **value** without adding any extra space or extra characters before or after.

QUESTION 5

```
# include <stdio.h>
# include <stdlib.h>
# include <sys/types.h>
# include <unistd.h>
int main(int argc, char *argv\[])
{
    pid_t pid;
    int m = 10;
    for (int i=0; i<3;i++){
       pid = fork();            // Line A
       if (pid ==0) {
            m*=i+1;
                  break;}
        if (pid == -1)
            exit(1);}
    if (pid == 0)
        printf ("%d ",m);        // Line B
    if (pid > 0) {
        wait(NULL);              // Line C
        printf ("Process 0 ");}
    return 0;
}
```

Note: Assume that Fork() always succeeds. Select all possible outputs:

- [x] 10 20 30 Process 0
- [ ] Process 0 10 20 30
- [x] 20 10 Process 0 30
- [x] 30 Process 0 20 10

QUESTION 6

Sixty percent of the code in a certain application is parallelizable. Apply Amdahl's Law to compute the speedup for running this application on:

- **dual-core** 1.4
- **quad-core CPU.** 1.8

Insert the value with **one digit** after the point, e.g 34.5

QUESTION 7

Trace the following program and answer the following questions:

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

int main(void)
{
      pid_t  pid;
      int    x = 6;
      pid = fork();
      if (pid == 0) {
            execl("/bin/ls", "ls", "-l", NULL);
            ++x;
            printf ("%d",x);
            exit(1); }
      else
            x = x * 2;
      return 0;
}
```

- What is the last value of x in the child process if execl() fails? 7
- What is the last value of x in the parent process if execl() fails? 12
- If execl() succeeds, Will the print statement be reached in the child process? (Yes/No) No
- What is the last value of x in the parent process if execl() succeeds? 12