1 Short questions: 3 marks each

1. Let a process **P** create a child process **P1**. Then, let **P1** create another child process **P2**. This way, the PPID (Parent Process ID) of **P1** is greater than the PPID of **P2**.
   Your answer(True/False) : False

2. How many values does the system call **fork()** return if successful?
   Your answer : 2

3. How many values does the system call **fork()** return if it fails?
   Your answer : 1

4. It is possible that several processes have the same *PPID*.
   Your answer(True/False) : True

5. What will be the PPID of a process, when its parent dies?
   Your answer : 1

6. **EOF** is a constant equal to -1.
   Your answer(True/False) : True

7. A process with no children will be blocked by the system call **wait()**.
   Your answer(True/False) : False

8. A signal typically triggers either a user-defined function or a default handler.
   Your answer(True/False) : true

9. When **exit()** fails, it returns -1 to the caller process.
   Your answer(True/False) : False

10. What will be printed by the statement below, assuming **fd** is a valid file descriptor:
    `printf("%d\n", (int)lseek(fd, 0L, SEEK_SET));`
    Your answer : 0

2 Tracing: 7 marks each

1. How many "Hi"s will this program print?

```
main(){
  for(int i=0; i < 4; i++)
    fork();

  printf("Hi\n");
}
```

   Your answer : **16**

2. How many "Hi"s will this program print?

```
main(){
  for(int i=0; i < 4; i++){
    fork();
    printf("Hi\n");
  }
}
```

Your answer : **30**

3. How many "Hi"s will this program print?

```
int main(){
  for(int i=0; i < 4; i++)
    if(fork())
      printf("Hi\n");
}
```

Your answer : **15**

4. What will be printed by the program below?

```
int main(void){
  if(!fork()){
    sleep(4);
    printf("Monday");
  }else{
    alarm(1);
    sleep(2);
    printf("November ");
    exit(0);
   }
}
```

Your answer : **Monday**

5. What will appear on the screen:

```
main(){
  int n=1;

  if(fork())
      wait(&n);
  else
      n=n+10;

  printf("%d\n", n);
  exit(0);
}
```

Your answer : **11 0**

6. What will appear on the screen when the following program runs?

```
int main(int argc, char *argv[]){
  printf("One ");
  fork();
  printf("Windsor\n");
  pause();
  printf("Two\n");
  exit(0);
}
```

Your answer :
One Windsor
One Windsor

7. What will be printed by the function below?

```
void myFunc(int fd, char *ar)){
  lseek(fd, 0L, SEEK_SET);
  write(fd, ar, 100);
  printf("%d\n", (int) lseek(fd, 0L, SEEK_CUR);
}
```

Your answer: 100

3 Coding(21 marks)

Modify the program below so that it acts like a server to flip images. In particular,

- the original process goes into an infinite loop,
- the original process requests an image name from user, then reads and opens it(if open fails, process should prints an error message and loops back to read the next image name).
- the original process reads the image header, then forks,
- the child process flips the image and exit with status 0
- the parent process waits for its child
- the parent process checks the return status of the child and if it is zero, it prints *image flipped* otherwise, it prints *unable to flip image*.

```
int main(int argc, char *argv[]){
 unsigned char buffer1[1000], buffer2[1000], line[255];
 int fd, i, nbcols, nblines, header=0;
 FILE *fp;
 if((fd = open(argv[1], O_RDWR)) == -1){
   perror("Opening file ");
   exit(1);
 }
 fp = fdopen(fd, "r"); // read header
 fgets(line, 255, fp);
 header+= strlen(line);
 fgets(line, 255, fp);
 header+= strlen(line);
 sscanf(line, "%d%d", &nblines, &nbcols);
 fgets(line, 255, fp);
 header+= strlen(line);
 for(i=0; i<nblines/2; i++){     // flipping image
   lseek(fd, (off_t)(header+i*nbcols), SEEK_SET);
   read(fd, buffer1, nbcols);
   lseek(fd, (off_t)(-(i+1)*nbcols), SEEK_END);
   read(fd, buffer2, nbcols);
   lseek(fd, (off_t)(header+i*nbcols), SEEK_SET);
   write(fd, buffer2, nbcols);
   lseek(fd, (off_t)(-(i+1)*nbcols), SEEK_END);
   write(fd, buffer1, nbcols);
 }
 close(fd);
 exit(0);
}
```

```
int main(int argc, char *argv[]){
 unsigned char buffer1[1000], buffer2[1000], line[255], imageName[30];;
 int fd, i, nbcols, nblines, header=0, status;
 FILE *fp;

 while(1){
   printf("Enter image to flip: ");
   scanf("%s", imageName);
   if((fd = open(imageName, O_RDWR)) == -1)
     printf(" Could not open %s \n", imageName);
   else{    // reading the header
     fp = fdopen(fd, "r");
     fgets(line, 255, fp);
     header+= strlen(line);
     fgets(line, 255, fp);
     header+= strlen(line);
     sscanf(line, "%d%d", &nblines, &nbcols);
     fgets(line, 255, fp);
     header+= strlen(line);

     if(!fork()){       // flipping the image by the child
        for(i=0; i<nblines/2; i++){
          lseek(fd, (off_t)(header+i*nbcols), SEEK_SET);
          read(fd, buffer1, nbcols);
          lseek(fd, (off_t)(-(i+1)*nbcols), SEEK_END);
          read(fd, buffer2, nbcols);

          lseek(fd, (off_t)(header+i*nbcols), SEEK_SET);
          write(fd, buffer2, nbcols);
          lseek(fd, (off_t)(-(i+1)*nbcols), SEEK_END);
          write(fd, buffer1, nbcols);
        }
        close(fd);
        exit(0);
     }
     wait(&status);
     if(status==0)
        printf("Image flipped successfully\n");
     else
        printf("Unable to flip image\n");
   }
 }
}
```