Each of these program prints a statement. Answer whether or not the statement is true.

Question 1:

```
int main()
{
  if (fork() == 0) {
    sleep(1);
    printf("I am an orphan process.\n");
  } else {
    sleep(2);
  }
  exit(0);
}
```

Question 2:

```
int main()
{
  int i = 5;

  if (fork() != 0) {
    i += 5;
    sleep(2);
  } else {
    sleep(1);
    printf("In my process, i equals 5.");
  }
}
```

Question 3:

```
int main()
{
  int fd;

  fd = open("f1.txt", O_WRONLY | O_CREAT, 0666);
  dup2(fd, 1);
  close(fd);

  if (fork() != 0) {
    dup2(2, 1);
    sleep(3);
  } else {
    sleep(1);
    printf("This line will print on the screen\n.");
  }
}
```

Question 4:

```
int main()
{
  int p[2];

  pipe(p);
  if (fork() == 0) {
    dup2(p[0], 0);
    printf("This process will never terminate.\n");
    execlp("cat", "cat", NULL);
    exit(0);
  } else {
    dup2(p[1], 1);
    close(0);
    sleep(1);
    exit(0);
  }
}
```

Question 5:

```
int main()
{
  if (fork() == 0) {
    sleep(2);
    printf("I am a zombie process.\n");
  } else {
    sleep(1);
  }
  exit(0);
}
```

Question 6:

```
int main()
{
  char *s = "ABCDEFGHIJKLMNOPQRSTUVWXYZ\n";
  int i;
  int p[2];

  pipe(p);
  close(p[0]);
  if (fork() == 0) {
    for (i = 2; i < 20; i++) dup2(p[1], i);
    printf("I am about to generate SIGPIPE.\n");
    while (1) write(10, s, strlen(s));
  }
}
```

Answer to Clicker Questions

Question 1

False -- the process would be an orphan if its parent exited before it exited. This is the other way around. The process is a zombie.

Question 2

True -- The child's address space is copied from the parent, so when the parent changes *i*, the child is unaffected.

Question 3

False -- in the child, stdout is going to **f1.txt**. The **dup2(2,1)** call in the parent does not affect the child.

Question 4

True -- the write end of the pipe is not closed in the child, so the **cat** process will never read EOF. Although the parent doesn't close the write end either, it exits, so it is irrelevant.

Question 5

False -- see question 1.

Question 6

True -- the **write(10)** calls will write to the write end of the pipe, and there is no read end. It may not happen on the first **write()** call, but it will happen on one of them.