Exceptional Control Flow Examples

Problem 1:

Suppose that the following C program is run on a singple processor machine.

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

int cnt = 0;

int main(void)
{
  if (fork()==0) {
    cnt++;
    fork();
    cnt++;
  }
  cnt++;
  printf("%d",cnt);
  return 0;
}
```

List all possible outputs of this program.

Problem 2:

This problem tests your understanding of Unix process control.
Consider the following C program. (For space reasons, we are not checking error return codes, so assume that all functions return normally.)

```
int main()
{
    int status;
    int counter = 1;

    if (fork() == 0) {
        counter++;
        printf("%d",counter);
    }
    else {
        if (fork() == 0) {
            printf("9");
            counter--;
            printf("%d",counter);
            exit(0);
        }
        else {
            if (wait(&status) > 0) {
               printf("6");
            }
        }
    }

    printf("8");
    exit(0);
}
```

For each of the following strings, circle whether (Y) or not (N) this string is a possible output of the program.

A. `298068` Y N

B. `291688` Y N

C. `920688` Y N

D. `268908` Y N

E. `906828` Y N

Problem 3:

onsider the following program:

```
int main()
{
  pid_t pid;

  if ((pid = fork()) == 0) {
    printf ("a ");
  }
  else {
    printf ("b ");
    if (fork() == 0) {
      printf ("c ");
    }
  }
  if (waitpid (pid, NULL, 0) > 0)
    printf ("d ");

  printf ("e ");
}
```

Which of the following are possible outputs of the program:

A. `a b c d e e e`

B. `b a e c d e e`

C. `a b c e e d e`

D. `b e c e a d e`

Problem 4:

Consider the following program which sends itself a delayed signal using the alarm function:

```
typedef void handler_t(int);

handler_t *Signal(int signum, handler_t *handler)
{
    struct sigaction action, old_action;
    action.sa_handler = handler;
    sigemptyset(&action.sa_mask);
    action.sa_flags = SA_RESTART;
    assert(sigaction(signum, &action, &old_action) >= 0);
    return (old_action.sa_handler);
}

void handler(int sig)
{
    static int beeps = 0;
    printf("YO\n");
    if (++beeps < 2)
        alarm(1); /* next SIGALRM will be delivered in 1s */
    else {
        printf("MA\n");
        kill(getpid(), SIGKILL);
    }
}

int main()
{
    Signal(SIGALRM, handler); /* install SIGALRM handler */
    alarm(1); /* next SIGALRM will be delivered in 1s */

    while (1)
         ;

    printf(" is Great!\n");
    return 0;
}
```

What does this program output?

Does this program terminate?

Problem 5:

Suppose we compile and run the following program:

```
#include <setjmp.h>
#include <stdlib.h>
#include <stdio.h>

jmp_buf stuff;
jmp_buf more_stuff;

int bar()
{
    if(setjmp(more_stuff) == 0)
        return 3;
    else
        return 6;
}

int foo()
{
    char c = getc(stdin);
    if(c == 'x')
        longjmp(stuff, 42);
    else if(c == 'y')
        longjmp(more_stuff,17);
    else
        return bar();
    return -1;
}

int main()
{
    int n;
    n = setjmp(stuff);

    while((n += foo()) < 0)
        sleep(1);

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

For each of the following input string, write what the output is. If the results are undefined, write a question mark.

1. `abc`
2. `xab`
3. `xxa`
4. `yab`
5. `xyz`