C test - Quest, Godalming.

Given 3 C programs (around 30-50 lines of code each), had to spot the mistakes in them and suggest fixes. The fixes didn't have to be elegant. Particular attention to be paid to mistakes that could slip through testing and cause occasional run-time failure. I can't remember the complete programs, but the following fragments give a flavour of the mistakes to be found:

```
#define ESC = 0x1b

char *getDate (void)

typedef struct
{
    char firstName [20];
    char *lastName;
} NAMES;

char *getDate (void)
{
    char date [30];
    time_t t;
    time (&t);
    strcpy (date, ctime (&t));
    return &date [0];
}

void allocNames (NAMES names)
{
    names.lastName
    if ((  = (char *) malloc (sizeof (char) * 20)) == 0)
    {
        print ("malloc failed \n");
        exit (1);
    }
}
```

C test - Quest, Godelming

```
void getNames (NAMES *names)
{   int i;
    allocNames (names);

    /* fill names with '?' */

    for (i = 0 ; i <= 19 ; i++);
    {
        names -> firstName [i] = '?';
        names -> lastName [i] = '?';
    }

    names -> firstName [i] = NULL;
    names -> lastName [i] = NULL;

    {
        char buf [50];

        sprintf (buf, "Your last name is %50s", getenv ("USER"));
        puts (buf);
        puts ("Please enter your first name");
        gets (buf);
    }   buf [19] = 0;
        strcpy (names -> firstName, buf);
    }
}

void main (void)
{
    NAMES names; int loop = 0;

    getNames (&names);

    while (loop = 0)
    {
        - —

    }
}
```

C++ test – Quest, Godalming

Test was optional – not to be answered if learning C++, since wrong answers would affect overall score. Just 4 multiple-choice questions:

1. A class member is to be accessible only by methods of the class and its sub-classes. Should it be declared:
   1. private
   2. public
   3. protected
   4. friend

2. A non-class method needs access to a private member of a class. How should the method be declared in the class:
   1. private
   2. static
   3. friend
   4. protected

3. If the constructor of a class is:
   ```
   Foo::Baz() {}
   ```
   the corresponding destructor will be:
   1. `~Foo::Baz() {}`
   2. `Foo::~Baz() {}`
   3. `~Baz() {}`
   4. – – –

4. If memory is allocated using:
   ```
   int a[] = new int[10];
   ```
   the memory should be freed using:
   1. `delete a;`
   2. `delete a[];`
   3. `delete[] a;`
   4. `delete a[10];`