Questions that appeared in past tests

0 **X:** Which of the following code snippets is a correct implementation of the **push** operation in a stack:

a)
```
void push(double v)
{
  assert(cur_size < max_size);
  data[cur_size++] = v;
}
```

b)
```
void push(double v)
{
  assert(cur_size > 0);
  data[--cur_size] = v;
}
```

c)
```
double push(void)
{
  assert(cur_size < max_size);
  return data[cur_size++];
}
```

d)
```
double push(void)
{
  assert(cur_size > 0);
  return data[--cur_size];
}
```

0 **X:** In a double-linked list, which of the following code snippets is a correct implementation of a function that counts the number of nodes located after the node given as argument to the function?

a)
```
int count_after(node *n)
{
  int i;

  for(i = 0;n->next != NULL;n = n->next)
    i++;
  return i;
}
```

b)
```
int count_after(node *n)
{
  int i;

  for(i = 0;n->next != NULL;n = n->prev)
    i++;
  return i;
}
```

c)
```
int count_after(node *n)
{
  int i;

  for(i = 0;n->prev != NULL;n = n->next)
    i++;
  return i;
}
```

d)
```
int count_after(node *n)
{
  int i;

  for(i = 0;n->prev != NULL;n = n->prev)
    i++;
  return i;
}
```

0 **X:** Which of the following functions can be used to increment an index in a circular buffer of size `size`?

a)
```
int inc_index(int i)
{
  return (i + 1 < size) ? i + 1 : 0;
}
```

b)
```
int inc_index(int i)
{
  return (i + 1 < size) ? 0 : i + 1;
}
```

c)
```
int inc_index(int i)
{
  return (i < size) ? i + 1 : 0;
}
```

d)
```
int inc_index(int i)
{
  return (i < size) ? i + 1 : size;
}
```

0 **X:** Explain what is a stack. Describe the operations it provides and what they do.

0 **X:** Explain how the information is organized in a max-heap.

0 **X:** Write on the left C++ code that implements the pop function (stack implemented as an array, data items are of type T), using some of the lines of code presented on the right.

```
                                        T pop(void)
                                        void pop(T v)
{                                       assert(cur_size > 0);
                                        assert(cur_size < max_size);
                                        return data[cur_size++];
                                        return data[cur_size--];
                                        return data[++cur_size];
                                        return data[--cur_size];
                                        data[cur_size++] = v;
}                                       data[cur_size--] = v;
                                        data[++cur_size] = v;
                                        data[--cur_size] = v;
```

Expect one question that asks for an explanation of a short piece of C code. It can be about a linked list, a stack, a queue, a min- or max-heap, or about an ordered or unordered binary tree.