Quiz 4 Version D

Due No due date	**Points** 20	**Questions** 13
Available after Nov 16 at 3:30pm		**Time Limit** None
Allowed Attempts Unlimited		

Instructions

Quiz for Lecture 4: const and Singly Linked Lists

18 points required to pass

Take the Quiz Again

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	less than 1 minute	0 out of 20 *

* Some questions not yet graded

Score for this attempt: **0** out of 20 *
Submitted Nov 28 at 2:10pm
This attempt took less than 1 minute.

Question 1

0 / 1 pts

Examine this code. Line numbers in main are specified in comments. In the check boxes below, check each line of code that WILL NOT COMPILE.

```
struct cl_node {   char data;   cl_node *next;   };

int main() {

    const double pi(3.141592654);         // 2
    const double e;                      // 3

    const cl_node n1{ 'a', nullptr };     // 4

    cl_node *p1 = &n1;                   // 5

    const cl_node *p2(&n1);              // 6

}
```

- [ ] Line 2 will NOT compile

orrect Answer

- [ ] Line 3 will NOT compile

'ou Answered

- [✓] Line 4 will NOT compile

n1 is a cl_node const, which must be initialized. We have initialized n1's data member (of type char) with 'a' and n1's next member (of type pointer-to cl_node) with nullptr, so this statement is correct and will compile.

orrect Answer

- [ ] Line 5 will NOT compile
- [ ] Line 6 will NOT compile

Jnanswered

Question 2

0 / 1 pts

Examine this code. Line numbers are specified in comments. In the check boxes below, check each line of code that WILL NOT COMPILE.

```
struct cl_node {   char data;   cl_node *next;   };

int main() {
    cl_node n1;                       // 2
    cl_node *p1{&n1};                 // 3
    const cl_node *p2;                // 4
    p2 = &n1;                         // 5
    p1->data = 'B';                   // 6
    *p1.next = 0;                     // 7
    p2->data = 'x';                   // 8
}
```

- ☐ Line 2 will NOT compile
- ☐ Line 3 will NOT compile
- ☐ Line 4 will NOT compile
- ☐ Line 5 will NOT compile
- ☐ Line 6 will NOT compile

orrect Answer

- ☐ Line 7 will NOT compile

orrect Answer

- ☐ Line 8 will NOT compile

Jnanswered

Question 3

0 / 1 pts

Suppose we have this structure definition, for a node in a singly linked list of **char**:

struct cl_node {   char data;   cl_node *next;   };

Which of these would be the most reasonable interface for a function that makes a singly linked list of **char** from a C-string?

- ○ **cl_node *mk_clist_from_Cstring(cl_node *p) { ... }**
- ○ **void mk_clist_from_Cstring(const char *cstr) { ... }**
- ○ **void mk_clist_from_Cstring(cl_node *p) { ... }**

orrect Answer

- ○ **cl_node *mk_clist_from_Cstring(const char *cstr) { ... }**
- ○ **cl_node **mk_clist_from_Cstring(const char *cstr) { ... }**
- ○ **char *mk_clist_from_Cstring(const char *cstr) { ... }**

Jnanswered

Question 4

0 / 1 pts

Suppose we have this structure definition, for a node in a singly linked list of **char**:

struct cl_node {   char data;   cl_node *next;   };

Which of these would be the most reasonable interface for a function that modifies the data in one or more of the linked list nodes, but does not modify the shape of the linked list?

orrect Answer

- ○ **void clist_mod_data(cl_node *p) { ... }**
- ○ **void clist_mod_data(cl_node **pp) { ... }**
- ○ **void clist_mod_data(const cl_node *p) { ... }**

Jnanswered

Question 5

0 / 1 pts

Suppose we have this structure definition, for a node in a singly linked list of **int**:

struct il_node { int data; il_node *next; };

Which of these would be the most reasonable interface for a function that accesses the data in one or more of the linked list nodes, but does not modify any data and does not modify the shape of the list?

- ○ **int ilist_access_data(il_node *p) { ... }**
- ○ **int ilist_access_data(il_node **pp) { ... }**

orrect Answer

- ○ **int ilist_access_data(const il_node *p) { ... }**

Jnanswered

Question 6

0 / 1 pts

Suppose we have this structure definition, for a node in a singly linked list of **double**:

struct dl_node { double data; dl_node *next; };

Which of these would be the most reasonable interface for a function that modifies the shape of the linked list (adding one or more nodes, deleting one or more nodes, changing the order of nodes, ...)?

- ○ **void dlist_change_shape(const dl_node *p) { ... }**
- ○ **void dlist_change_shape(dl_node *p) { ... }**

orrect Answer

○ **void dlist_change_shape(dl_node **pp) { ... }**

Question 7

Not yet graded / 2 pts

Suppose we have this structure definition, for a node in a singly linked list of **int**:

struct il_node { int data; il_node *next; };

Write the body of code for this function, that returns the number of ***odd numbers*** in the list of integers:

```
size_t ilist_count_odd(const il_node *p)
{
    // this is the part you have to write as your answer
}
```

Your Answer:

```
size_t nodd(0);
for ( ; p; p = p->next)
   if (p->data % 2)  // odd is 1 (true), even is 0 (false)
      nodd += 1;
return nodd;
```

Question 8

Not yet graded / 2 pts

Suppose we have this structure definition, for a node in a singly linked list of **int**:

struct il_node {   int data;   il_node *next;   };

Write the body of code for this function, that returns **true** if all the data values in the list are > 0, and **false** otherwise:

bool ilist_all_gt_0(const il_node *p)

{

    // this is the part you have to write as your answer

}

Your Answer:

```
for ( ; p; p = p->next)
   if (p->data <= 0)
      return false;
return true;
```

Question 9

Not yet graded / 2 pts

Suppose we have this structure definition, for a node in a singly linked list of **char**:

struct cl_node {   char data;   cl_node *next;   };

Write the body of code for this function, that creates a list of **n** copies of the character in parameter **c**, and returns a pointer to the first node in the list:

```
cl_node *mk_clist_from_n_c(char c, size_t n)
{
    // this is the part you have to write as your answer
}
```

Your Answer:

```
// an iterative implementation
cl_node *first(nullptr);
cl_node *last(nullptr);
for ( ; n; --n) {
    cl_node *pnew = new cl_node{ c, nullptr };
    if (!first)
        first = last = pnew;
    else
        last = last->next = pnew;
}
return first;
```

Question 10

Not yet graded / 2 pts

Suppose we have this structure definition, for a node in a singly linked list of **double**:

struct dl_node { double data; dl_node *next; };

Write the body of code for this function, that replaces the current **data** value of each node with its inverse (that is, 1 / **data**). If **data** is **0.0**, replace **data** with **DBL_MAX** (don't worry about specifying the header).

void dlist_invert_values(dl_node *p)

{

// this is the part you have to write as your answer

}

Your Answer:

```
for ( ; p; p = p->next)
    if (p->data == 0.0)
        p->data = DBL_MAX;
    else
        p->data = 1 / p->data;
```

Question 11

Not yet graded / 2 pts

Suppose we have this structure definition, for a node in a singly linked list of **char**:

struct cl_node { char data; cl_node *next; };

Write the body of code for this function, that creates an ***empty*** list of characters, and returns an appropriate value for the header node.

cl_node *mk_empty_clist()

{

// this is the part you have to write as your answer

}

Your Answer:

return nullptr;

Question 12

Not yet graded / 2 pts

Suppose we have this structure definition, for a node in a singly linked list of **double**:

struct dl_node { double data; dl_node *next; };

Write the body of code for this function, that displays each value in the list in a field 12 characters wide (don't worry about any header(s) you might need):

void dlist_display_pretty(const dl_node *p)

{

// this is the part you have to write as your answer

}

Your Answer:

```
for ( ; p; p = p->next)
    cout << setw(12) << p->data;
```

Question 13

Not yet graded / 2 pts

Suppose we have this structure definition, for a node in a singly linked list of **double**:

struct dl_node { double data; dl_node *next; };

Write the body of code for this function, that moves the last node in the list to the front of the list. ***Please note: You may not delete any nodes or allocate any new nodes: you may only move existing nodes, as needed.*** If the list is empty or only has one node, this function should do nothing.

```
void dlist_rotate_minus_one(dl_node **pp)
{
    // this is the part you have to write as your answer
}
```

Your Answer:

```
// do nothing if 0 or 1 nodes

if (*pp == nullptr || (*pp)->next == nullptr)

   return;

// find the next-to-last node

dl_node *next_to_last(*pp);

for ( ; next_to_last->next->next; next_to_last = next_to_last->next)

   ;

// make the last node the first node

next_to_last->next->next = *pp;

*pp = next_to_last->next

// make the former next-to-last node the last node

next_to_last->next = nullptr;
```

Quiz Score: **0** out of 20