Quiz 4 Version E

Due No due date **Points** 20 **Questions** 13
Available after Nov 18 at 1pm **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 cl_node n1{ 'a', nullptr };        // 2

     cl_node *p1 = &n1;                       // 3
```

```
    const cl_node *p2(&n1);                   // 4
    cl_node *p3 = new cl_node{ 'b', nullptr };     // 5
    const cl_node *p4{p3};                    // 6
    cout << p4->data << '\n';                 // 7
    n1.data = 'z';                            // 8
}
```

'ou Answered

[✓ Line 2 will NOT compile]

orrect Answer

☐ Line 3 will NOT compile

☐ Line 4 will NOT compile

☐ Line 5 will NOT compile

☐ Line 6 will NOT compile

☐ Line 7 will NOT compile

orrect Answer

☐ Line 8 will NOT compile

Jnanswered

Question 2

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?

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

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 modifies the data in one or more of the linked list nodes, but does not modify the shape of the linked list?

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

orrect Answer

○ **void clist_mod_data(cl_node *p) { ... }**

Jnanswered

Question 4

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?

orrect Answer

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

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

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

Jnanswered

Question 5

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, ...)?

orrect Answer

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

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

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

Jnanswered

Question 6

0 / 1 pts

What is typically specified as the return type of a linked list interface function, if the value could be any non-negative integer?

○ int

○ void *

○ sizeof

orrect Answer

○ **size_t**

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 nodes in the list of integers:

```
size_t ilist_size(const il_node *p)

{

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

}
```

Your Answer:

```
size_t sz(0);

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

    sz += 1;

return sz;
```

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 odd integers, and **false** otherwise:

```
bool ilist_all_odd(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 % 2 == 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 returns the address of the first **cl_node** in a list of **char** that is a duplicate of the argument list of **char**.

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

Your Answer:

```
// first version
cl_node *first(nullptr);
cl_node *last(nullptr);
for ( ; p; p = p->next) {
    cl_node *pnew = new cl_node{ p->data, nullptr };
    if (!last) {
        first = last = pnew;
    }
    else {
        last = last->next = pnew;
    }
}
return first;


// second version
cl_node *p2(0);
cl_node **pp2(&p2);
for ( ; p; p = p->next) {
    *pp2 = new cl_node{ p->data, nullptr };
    pp2 = &(*pp2)->next;
}
return p2;
```

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 increments (that is, adds **1** to) the current **data** value of each node.

void dlist_increment(dl_node *p)

{

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

}

Your Answer:

```
for ( ; p; p = p->next)
    p->data += 1;
// or p->data++, ++p->data, p->data = p->data + 1
```

Question 11

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 deletes the second node from the list of **double**. If the list has fewer than two nodes, this function should do nothing.

void dlist_delete_second(dl_node **pp)

{

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

}

Your Answer:

```
if (*pp == nullptr || (*pp)->next == nullptr)
    return;
dl_node *tmp = (*pp)->next;
(*pp)->next = (*pp)->next->next;
delete tmp;
```

Question 12

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 a pointer to the first **il_node** in a list of **int** values, where the values count down from **N** to and including **0**. For example, if **N** is **5**, the list should contain the values **5**, **4**, **3**, **2**, **1**, **0**. If **N** is negative, the function should do nothing.

```
il_node *mk_ilist_countdown(int N)

{

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

}
```

Your Answer:

```
// an iterative version

il_node *first(nullptr);

il_node *last(nullptr);

for ( ; N >= 0; --N) {

    il_node *pnew = new il_node{ N, nullptr };

    if (!last)

        last = first = pnew;

    else

        last = last->next = pnew;

}

return first;


// a recursive version

if (N < 0)

    return nullptr;

return new il_node{ N, mk_ilist_countdown(N - 1) };
```

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 displays each value in the list to **cout** in a field **12** characters wide.

```
void dlist_display_12_wide(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;
```

Quiz Score: **0** out of 20