Data Structures
Fall 2020
Prof. George Wolberg
Midterm (Part 2: Chap. 4-9)

CHAPTER 4 PROBLEMS

Problem 4.1. (1 point)

Suppose that p is an int* variable. Write several lines of code that will make p point to an array of 100 integers, place the numbers 0 through 99 into the array components, and return the array to the heap. Do not use malloc.

Problem 4.2. (1 point)

Consider the following prototype:

```
function foo (const int * p);
```

What restriction does the const keyword provide within the implementation of foo?

Problem 4.3. (1 point)

Chapter 4 provides a bag class that stores the items in a dynamic array. Use an example with pictures to explain what goes wrong if the automatic assignment operator is used for this bag.

Problem 4.4. (1 point)

Consider the bag from Chapter 4 (using a dynamic array). If the insert function is activated, and the bag is full, then the bag is resized with the statement:

```
reserve(used + 1);
```

Describe a problem with this approach, and provide a better solution.

Problem 4.5. (1 point)

Consider the following statements:

```
int i = 42;
int j = 80;
int *p1;
int *p2;
p1 = &i;
p2 = &j;
*p1 = *p2;
cout << i << j << endl;
```

What numbers are printed by the output statement?

A. 42 and then another 42
B. 42 and then 80
C. 80 and then 42
D. 80 and then another 80

Problem 4.6. (1 point)

What is printed by these statements?

```
int i = 1;
int k = 2;
int *p1;
int *p2;
p1 = &i;
p2 = &k;
p1 = p2;
*p1 = 3;
*p2 = 4;
cout << i;
```

A. 1
B. 2
C. 3
D. 4

Problem 4.7. (1 point)

When should a pointer parameter p be a reference parameter?
A. When the function changes p, and you want the change to affect the actual pointer argument.
B. When the function changes p, and you do NOT want the change to affect the actual pointer argument.
C. When the function changes *p, and you want the change to affect the the object that is pointed at.
D. When the function changes *p, and you do NOT want the change to affect the the object that is pointed at.
E. When the pointer points to a large object.

Problem 4.8. (1 point)

Suppose that you want to declare an array of characters to hold a C++ string with exactly 9 letters. Which declaration is best?
A. char s[8];
B. char s[9];
C. char s[10];
D. char s[11];
E. char s[12];

Problem 4.9. (1 point)

Suppose that x and y are two C++ strings. Which expression will return true whenever x and y contain the same sequence of characters?
A. (x = y)
B. (x == y)
C. (x != y)
D. (strcmp(x, y))

CHAPTER 5 PROBLEMS

Problem 5.1. (1 point)

Suppose cursor points to a node in a linked list (using the node definition with member functions called data and link). What statement changes cursor so that it points to the next node?
A. cursor++;
B. cursor = link( );
C. cursor += link( );
D. cursor = cursor->link( );

Problem 5.2. (1 point)

Suppose cursor points to a node in a linked list (using the node definition with member functions called data and link). What Boolean expression will be true when cursor points to the tail node of the list?
A. (cursor == NULL)
B. (cursor->link( ) == NULL)
C. (cursor->data( ) == NULL)
D. (cursor->data( ) == 0.0)
E. None of the above.

Problem 5.3. (1 point)

Suppose that p is a pointer variable that contains the NULL pointer. What happens if your program tries to read or write *p?
A. A syntax error always occurs at compilation time.
B. A run-time error always occurs when *p is evaluated.
C. A run-time error always occurs when the program finishes.
D. The results are unpredictable.

Problem 5.4. (1 point)

Suppose that f is a function with a prototype like this:

```
void f(________ head_ptr);
// Precondition: head_ptr is a head pointer for a linked list.
// Postcondition: The function f has done some computation with the linked list, but the list itself is unchanged.
```

What is the best data type for head_ptr in this function?
A. node
B. const node
C. node*
D. const node*

Problem 5.5. (1 point)

Suppose that f is a function with a prototype like this:

```
void f(________ head_ptr);
// Precondition: head_ptr is a head pointer for a linked list.
// Postcondition: The function f has done some manipulation of
// the linked list, and the list might now have a new head node.
```

What is the best data type for head_ptr in this function?
A. node
B. node&
C. node*
D. node*&

Problem 5.6. (1 point)

What kind of list is best to answer questions such as "What is the item at position n?"
A. Lists implemented with an array.
B. Doubly-linked lists.
C. Singly-linked lists.
D. Doubly-linked or singly-linked lists are equally best

CHAPTER 6 PROBLEMS

Problem 6.1. (1 point)

Write one or two short sentences to clearly explain the advantage of template functions.

Problem 6.2. (1 point)

Consider the code below which is supposed to shift the values from data[0] through data[n-1] rightward one position (so these values now reside in data[1] through data[n]).

```
size_t i;
for (i = 0; i < n; ++i)
   data[i+1] = data[i];
```

There is a bug in the above code. Write one sentence to describe the bug and write new code that does the job correctly.

Problem 6.3. (1 point)

Complete the body of this template function. Check the precondition as much as possible, and don't cause a heap leak.

```
template <class Item>
void list_head_remove(node<Item>*& head_ptr)
// Precondition: head_ptr is the head pointer of a linked list,
// with at least one node.
// Postcondition: The head node has been removed and returned to the heap;
// head_ptr is now the head pointer of the new, shorter linked list.
```

Problem 6.4. (1 point)

What is the primary purpose of template functions?
A. To allow a single function to be used with varying types of arguments
B. To hide the name of the function from the linker (preventing duplicate symbols)
C. To implement container classes
D. To permit the use of the debugger without the -gstabs flag

Problem 6.5. (1 point)

When should a function be implemented as a template function?
A. When the data types of the parameters all have copy constructors.
B. When the function depends on an underlying data type.
C. When the function is relatively short (usually just one line).
D. When the function only takes one argument.

Problem 6.6. (1 point)

What is a major difference between the header file for a toolkit of template functions and the header file for a toolkit of ordinary functions?
A. The ordinary function toolkit header file must have a #include for the implementation file, but the template version does not have this #include.
B. The template function toolkit header file must have a #include for the implementation file, but the ordinary version does not have this #include.
C. The ordinary function toolkit header file must have a macro guard, but the template version does not have this macro guard.
D. The template function toolkit header file must have a macro guard, but the ordinary version does not have this macro guard.

Problem 6.7. (1 point)

Suppose bag is a template class, what is the syntax for declaring a bag b of integers?
A. bag b;
B. bag<int> b;
C. bag of int b;
D. int bag b;

CHAPTER 7 PROBLEMS

Problem 7.1. (1 point)

Consider the following pseudocode:

```
declare a stack of characters
while ( there are more characters in the word to read )
{
  read a character
  push the character on the stack
}
while ( the stack is not empty )
{
  write the stack's top character to the screen
  pop a character off the stack
}
```

What is written to the screen for the input "carpets"?
A. serc
B. carpets
C. steprac
D. ccaarrppeettss

Problem 7.2. (1 point)

Here is an INCORRECT pseudocode for the algorithm which is supposed to determine whether a sequence of parentheses is balanced:

```
declare a character stack
while ( more input is available)
{
  read a character
  if ( the character is a '(' )
    push it on the stack
  else if ( the character is a ')' and the stack is not empty )
    pop a character off the stack
  else
    print "unbalanced" and exit
}
print "balanced"
```

Which of these unbalanced sequences does the above code think is balanced?
A. ((())
B. ())(()
C. (()()))
D. ((()))()

Problem 7.3. (1 point)

Suppose we have an array implementation of the stack class, with ten items in the stack stored at data[0] through data[9]. The CAPACITY is 42. Where does the push member function place the new entry in the array?
A. data[0]
B. data[1]
C. data[9]
D. data[10]

CHAPTER 8 PROBLEMS

Problem 8.1. (1 point)

One difference between a queue and a stack is:
A. Queues require dynamic memory, but stacks do not.
B. Stacks require dynamic memory, but queues do not.
C. Queues use two ends of the structure; stacks use only one.
D. Stacks use two ends of the structure, queues use only one.

Problem 8.2. (1 point)

If the characters 'D', 'C', 'B', 'A' are placed in a queue (in that order), and then removed one at a time, in what order will they be removed?
A. ABCD
B. ABDC
C. DCAB
D. DCBA

CHAPTER 9 PROBLEMS

Problem 9.1. (1 point)

Write a recursive function that has one parameter which is a size_t value called x. The function prints x asterisks, followed by x exclamation points. Do NOT use any loops. Do NOT use any variables other than x.

Problem 9.2. (1 point)

Implement the following function. Do not use any local variables or loops.

```
void pattern(unsigned int n)
// Precondition: n > 0;
// Postcondition: The output consists of lines of integers. The first line
// is the number n. The next line is the number 2n. The next line is
// the number 4n, and so on until you reach a number that is larger than
// 4242. This list of numbers is then repeated backward until you get back
// to n.
/* Example output with n = 840:
840
1680
3360
6720
6720
3360
1680
840
```

Problem 9.3. (1 point)

When the compiler compiles your program, how is a recursive call treated differently than a non-recursive function call?

A. Parameters are all treated as reference arguments
B. Parameters are all treated as value arguments
C. There is no duplication of local variables
D. None of the above

Problem 9.4. (1 point)

Consider the following function:

```
void test_a(int n)
{
  cout << n << " ";
  if (n>0)
    test_a(n-2);
}
```

What is printed by the call test_a(4)?

A. 0 2 4
B. 0 2
C. 2 4
D. 4 2
E. 4 2 0