==========

OLD OLD CS211 final exam Yoshii – try all Qs up to Section 5

==========

==========

Section 1. Fundamentals of C++: Review of Structures From Week 2

==========

1.] I want a structure called show which has two fields:
- the title (e.g. XFactor)
- the channel (e.g. 10).
To declare this structure type called show, I would write:

a. struct show
{ char title;
int channel; }

b. struct show
{ char title;
int channel; };

c. struct show
{ string title;
int channel; }

d. struct show
{ string title;
int channel; };

2.] To declare an array called NewShows such that each slot will contain a show structure, I would write:
(note: NewShows array has 12 slots)

a. NewShows show[12];
b. show NewShows[11];
c. show NewShows[12];
d. struct NewShows[11];
e. struct NewShows[12];

3.] To fill the NewShows array with information obtained interactively from the user, I would use the following loop.
(note: The user always has 12 shows)

```
for (int i = 0; i < 12; i++)
{
   cout << "Enter a show's title: ";
   <?>
   cout << "Enter its channel number: ";
   <????>
```

```
}
```

What should replace <?> and <????> ?

a. <?> is cin >> NewShows.title;
<????> is cin >> NewShows.channel;

b. <?> is cin >> title;
<????> is cin >> channel;

c. <?> is cin >> NewShows[i].title;
<????> is cin >> NewShows[i].channel;

d. <?> is cin >> NewShows.title[i];
<????> is cin >> NewShows.channel[i];

4.] To search through the above array (NewShows) to find the first slot whose show's title is RIKA, and display its slot number, I would use the following loop.

```
Found = 0;  // we have not found it yet so the flag is down
for (int i = 0; <?>; i++)
{
 if ( // the title is RIKA )
  {  cout << "Found it in slot" << i << endl;
     Found = 1;  // stop the loop by raising the flag
  }
}
```

What should be the conditional <?> to stop the loop as soon as it is found ?

a. (i < 12) && (Found == 1)
b. (i < 12) || (Found == 1)
c. (i < 12) && (Found == 0)
d. (i < 12) || (Found == 0)

==========

Section 2. Basics of Abstract Data Types From Week 4

==========

5.] What are the features of Abstract Data Types?

a. Hide details of data members from the clients
b. Hide implementation of member functions from the clients
c. Hide names of all data members from the clients
d. Hide names of all member functions from the clients
e. a, b and c are true.

Let's say you want to declare a class called <u>show</u>:
Its data members are title and channel.

6.] What are the parts of its class definition in the show.h (header) file? (answer based on the cs211 assignments)

a. data member declaration
b. member function prototypes
c. member function implementation
d. #include "show.C"
e. both a and b are true

7.] To declare its member function called Get_Title in the file show.C, I would start with:

a. string show_Get_Title()
b. string show.Get_Title()
c. string Get_Title()
d. string show::Get_Title()

8.] How does the client declare a show object called MyShow?

a. MyShow show;
b. show MyShow;
c. New Show = MyShow;
d. MyShow = New show;

9.] How does the client call Get_Title with MyShow as the object?

a. MyShow show::Get_Title()
b. Get_Title(MyShow)
c. MyShow::Get_Title()
d. MyShow.Get_Title()

10.] Which goals of software engineering do ADTs help us reach?

a. information hiding
b. loose coupling
c. cohesiveness
d. all of the above
e. only a and c

==========

Section 3. Stack From Week 5 and 6

==========

11.] For element type, we did typedef int el_t; and used el_t instead of int. Why?

a. Because in header files, int is not allowed
b. Because int has a special meaning when used with stacks
c. Because if we want to change the element type, we will need to change it in only one place

12.] In member function definitions,
stackError was called for error handling because

a. public functions Pop and Push are not allowed to do exit(1)
b. easier to modify error handling if it is isolated in one function
c. every ADT is required to have one private member function
d. all of the above

IMPORTANT: For this section, the data members are STK and Top;
And, Top indicates the slot where the top element is in STK.
Thus, if Top is -1, then the STK is empty.

13.] Which is the correct implementation of Pop(el_t& e)
(isEmpty has been checked already)

a. Top--;
e = STK[Top];

b. Top--;
e = STK[Top-1];

c. e = STK[Top];
Top--;

d. e = STK[Top-1];
Top--;

14.] Which is the correct sequence of statements for Push(el_t e)?
(isFull has been checked already)

a. Top++;
STK[Top] = e;

b. STK[Top] = e;
Top++;

c. STK[Top] = e;

d. STK[MAX] = e;

15.] An example of an application area for stacks is:

a. Simulation of planes waiting to take off.
b. Postfix expression evaluation.
c. Matching parentheses.

d. Traversing trees.
e. Some of the above but not all of the above.

==========

Section 4 Queue (Array Implementation) From Week 7

==========

IMPORTANT: For these problems, the data members are Q, Front and Rear.
Front is the index for the front element. (where the front element is)
Rear is the index for the rear element. (where the rear element is)
All questions in this section refer to an array implementation.

16.] Which is the correct sequence of statements for function remove(el_t& e)?
(isEmpty has been checked already)

a. e = Q[Front];
Front++;

b. e = Q[Front];
Front = (Front+1) % QUEUE_SIZE;

c. Front++;
e = Q[Front];

d. Front = (Front+1) % QUEUE_SIZE;
e = Q[Front];

17.] Which is the correct sequence of statements for function add(el_t e)?
(isFull has been checked already)

a. Q[Rear] = e;
Rear++;

b. Q[Rear] = e;
Rear = (Rear+1) % QUEUE_SIZE;

c. Rear++;
Q[Rear] = e;

d. Rear = (Rear+1) % QUEUE_SIZE;
Q[Rear] = e;

==========

Section 5 Pointers From Week 8 and 9

==========

18.] How do you declare a pointer variable P which will point to a cell

containing an integer?

a. int P;
b. int *P;
c. int P->;

19.] How do you then make P point to a brand new cell??
a. P = new int;
b. new P;
c. *P = new int;

20.] How do you display what's in the cell pointed to by P?

a. cout << P;
b. cout << *P;
c. cout << ->P;

21.] How do you then make P point to the same place as Q, leaving no garbage behind? SUGGESTION: Draw pictures!!

a. delete P;
P = Q;
b. delete P;
*P = *Q;
c. delete Q;
P = Q;
d. delete Q
Q = P;

22.] P and Q are pointing to the same cell. R is pointing elsewhere. How do you end up creating a dangling reference/pointer? SUGGESTION: Draw pictures!!

a. delete P;
b. delete *P;
c. P = R;
d. R = P;

==========

Section 6 Linked Lists (with Front and Rear pointers) From Week 9 and 10

==========

IMPORTANT: For this section, Front points to the first node, and Rear points to the last node.
SUGGESTION: Draw pictures for each choice!!!!!!!!!!

23.] The linked list currently has 5 nodes. How do you add a new node to the end?

a. Rear = new Node;
   Rear->Next = NULL;

b. Rear->Next = new Node;
   Rear = Rear->Next;
   Rear->Next = NULL;

c. Rear = new Node;
   Rear->Next = Rear;
   Rear->Next = NULL;

24.] How do you add a node to the front of the list?

a. Front = new Node;

b. NewFront = new Node;
   NewFront->Next = Front;
   Front = NewFront;

c. NewFront = new Node;
   Front = NewFront;
   Front->Next = NewFront;

25.] I want to insert a node right after the node pointed to by P. What is wrong with the following?

```
N = new node;
P->next = N;
N->next = P->next;
```

a. nothing
b. N->next will point to N itself
c. the node following N will point back to N

26.] How do you move the P pointer to move to the node previous to the rear node (i.e. the one right before the rear node)?

a. while (P != Rear) P = P->Next;
b. while (P->Next != Rear) P = P->Next;
c. while (P->Next != NULL) P = P->Next;

==========

Section 7 Advanced Abstract Data Types From Week 11

==========

27.] What does this-> refer to?

a. An object passed to a function as a parameter.
b. The receiver object which is the first object created by the client.

c. The receiver object which is what the client specified in front of the function name.
d. All of the above.

28.] Operator overloading allows us:

a. to change the meaning of an operator for int, char and float.
b. to change a binary operator into a unary one.
c. to use existing operators with new types of objects.

29.] Overloading of the operator = is required:

a. for all ADT's we write
b. when objects have pointers (e.g. linked lists) since
the default = operator the copies pointers, too.
c. when objects have pointers (e.g. linked lists) since = is undefined for such things.

==========

Section 8 Trees From Week 12

==========

IMPORTANT: Here, assume that the root is at level 0.
If you have only the root, the tree has just 1 level.
2^L nodes are at level L.

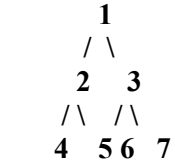
30.] With a binary tree in which every internal node has exactly 2 children and leaves are all at the same level, how many nodes are there in the tree if the bottom level is level B?

a. 2^B
b. 2^(B+1)
c. 2^(B+1) - 1

31.] With a binary tree in which every internal node has exactly 2 children and leaves are all at the same level, how many different levels are there? (N nodes are in the tree)

a. log (N-1) (base 2)
b. log N (base 2)
c. log (N+1) (base 2)
d. N

32.] Given the following tree, what is the depth first traversal order?



a. 1 2 3 4 5 6 7

b. 1 2 4 5 3 6 7
c. 4 5 6 7 2 3 1
d. 4 5 2 1 6 7 3

==========

Section 9 Complexity Analysis From Week 13

==========

33.] To determine the time complexity of an algorithm, you would count/measure:

a. CPU time on the fastest PC
b. repeated key operations such as comparison
c. number of lines of the C++ code
d. number of loops in the C++ code

34.] In Binary Search through an array containing N numbers, the O-notation for the worst case is:

a. O(N^2)
b. O(N)
c. O(log N) base 2
d. O(N/2)

35.] In Selection Sort of an array containing N numbers, the O-notation for the best case is:

a. O(N^2)
b. O(N)
c. O(log N) base 2
d. O(N/2)

36.] In Bubble Sort of an array containing N numbers, The O-notation for the best case is:

a. O(N^2)
b. O(N)
c. O(log N) base 2
d. O(N/2)

37.] In O-notations, we ignore lesser terms and constants because:

a. All we want is an approximate average number
b. We are interested in the growth rate as N gets large
c. It is impossible to figure out the lesser terms

That's it. You are done!!!!! Read over your answers carefully.