速成教育
SPEED UP EDUCATION

CSCA48

Introduction to Computer Science II

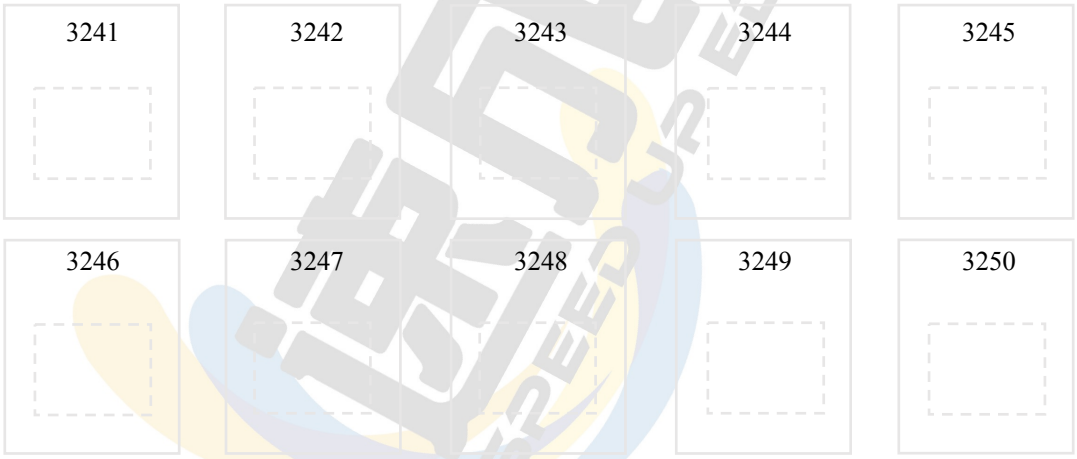
导师： VC

2022W CSCA48 Final Exam

Unit 1&2 – Memory model and C programming

1. [5 marks] Consider carefully the following little program and show in the memory diagram below what varaiables are allocated and their final value.

```
01  int main()
02  {
03      int values[3] = {5, 3, 15};
04      int *ptrs[2];
05
06      ptrs[0] = &values[0];
07      ptrs[1] = &values[2];
08      *(ptrs[0]) = 22;
09      ptrs[0] = ptrs[0] + 1;
10      *(ptrs[0] + 1) = 3;
11      *(ptrs[1] - 1) = 289;
12      *(ptrs[2]) = 0;                          // <- a) in question 2
13
14      ptrs[1] = (int *) calloc(5, sizeof(int));    // you get locker # 2211
15      *(ptrs[1] + 5) = 3415;                   // <- b) in question 2
16
17      // c) - f) below, suppose we add the corresponding line below
18      // this comment (only one for each question , not all of them).
19
20      return 0;
21  }
```



2. [3 marks] Circle in the space below any instructions that will likely cause a segfault (program crash) given the code from 1). You get .5 for each correctly circled answer, and .5 for each correctly not-circled answer! Assume the instructions below are independent from each other (not carried out in sequence!).

a) (line #12 above)
b) (line #15 above)
c) free(ptrs[1])
d) free(&ptrs[1]);
e) free(*(ptrs[1]));
f) ptrs[0] = ptrs[0] + 2561;

Unit 3 – Organizing, Storing, and Accessing Data

Suppose we're implementing the basic drawing functionality of something like Power Point – we will keep a linked list of objects to he drawn, they are stored using the CDT shown below.

```
typedef struct simple_object
{
     double Position[2];// Position [x, y] of this object in the scene
     double Colour[3];  // RGB colour of the object
     int depth;              // 'depth' at which to draw this (objects can overlap)
     struct simple_object *next;
} Object;
```

Suppose we already implemented all the basic functionality of a linked list to maintain the object data – that means we have the following functions: **serach(), insert(),** and **delete()** (don't foreget you have these!).

3. [3 marks] Let's now add a function to **modify** an object from the list, for instance, if the use wanted to move it around or change its coloiur, or change the order in which it is drawn to bring it forward or send it backward. In the space below, provide the code for a function that can be used to **modify the variables that define the object** (the user may want to update any/all of these variables). Assume that we **have a pointer to the specific object we want to updaste** (e.g, the clicked on it on screen, and we went and found where this object is in the linked list). **The function should NOT change the object type.**

```
            modify(                                                              )
{




}
```

4. [3 marks] A common issue with drawing problems after the use has been copying/pasting things around is that there will be duplicate objects – we want to provide a function to detect and remove duplicated objects. In the space below, implement a function that finds and removes the duplicates. Assume you have a helper function **checkForDup()** which takes pointers to two objects, and returns 1 if they are duplicated of each other.

```
        removeDups( )
{




}
```

Unit 4 – Cmplexoty, Sorting, and BSTs

5. [2 marks] What is the worst case big O complexity of removeDups() for a huge list of N objects. Provide s short justification for your answer. Assume checkForDup() is O(1).

6. [2 marks] You are writing a little app to let users suggest song to each other based on similar tastes. One of the function in your app simply checks whether two music collections have the same albums. **If the collections are stored in linked lists**, what is the **worst-case** complexity of this function? Albums can have completely different ordering in each list. **Explain your reasoning and give the Big O complexity.**

7. [3 marks] Suppose that your decided it's worth using a more clever data structure and you implement the collections using **BSTs**, using a **key** that depends on the album's title, artist, and year. What is the **worst-case** complexity of the function that checks whether the lists have the same albums? **Explain your reasongin and provide the BigO complexity.**

8. [5 marks] (crunky) – Suppose we have a BST stroing important information and we want to make sure that operations on the BST remain efficient. We want to check when the BST is becoming unbalanced there are very long branches and very short branches, which reduces the effectiveness of the BST operations. The first step to keep a BST balanced is to determine when it's becoming unbalanced. To do this, we will write a function **checkForBalance()** what will return **1 if the BST is balanced, and 0 otherwise**. For this function: **A BST is balanced if the height of the leaves changes by at most one** – another way to think of this: the minimum height of any leaf in the tree, and the maximum height of any leaf in the tree must not differ by more than 1 for the BST to be balanced. **Provide pseudocode for the function, and a BigO estimate of the complexity of checking for balance.**

9. [2 marks] Show the BST that results from inserting 15, 5, 27, 37, 47, 42, 1, 3, 4, 89, 31 in that order

10. [2 marks] Show the BST that result from deleting 37 from the tree in (9).

11. [2 marks] Show the BST that results from deleting 15 from the tree in (10).

12. [2 marks] What is the result of printing a post-order traversal of the tree in (11)?

Unit 5 – Graphs and Recursion

13. [2 marks] Show a graph below representing a problem for which it makes sense to have weighted edges. Explain why the graph makes sense for the corresponding problem and why edges must be weigthed.

14. [2 marks] Suppose we are writing an application for hospital, that application represents every possible type of medicine that could be given to a patient as a node, and the edges represent **negative interactions** between medicines (i.e. if an edge exist between medicine **A** and medicine **B**, then these two can not be given together because they combine to produce negative side effects). Suppose a doctor is aboue to prescribe medicine **A** for a patient. What is the complexity of checking **which medicines should not be taken together with medicine A if the graph is kept in an adjacency list?** the graph has N=|V| nodes and |E1| edges.
Give a BigO estimate and explain your reasoning.

15. [3 marks] Now suppose we have another graph in which medicines **A** and **B** have an edge joining them if both medicines have the same function (i.e. they can be substituted for one another to treat some specific medical issue). **Using both the graph that describes which medicines can't be taken together; and the graph that indicates which medicines are substitutes** give the worst-case BigO complexity of determining a suitable substitute **C** for a given durg **B** that can't be taken by a patient who is already on medicine **A**. both graphs have the same number of nodes. This graph has N=|V| nodes and |E2| edges**. Both graphs are kept as adjacency lists.**
Give a short pseudocode for the process to solve this task, as well as the BigO complexity.

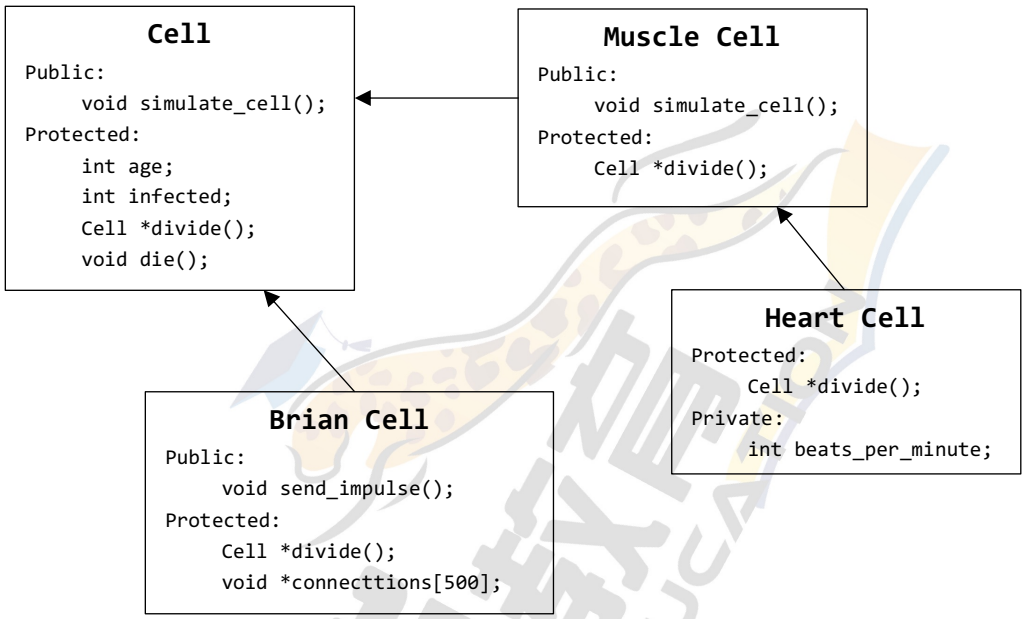
16. [3 marks] What is the worst-case BigO complexity of solving the same problem as (15) if both graphs are kept as **adjacency matrixes?**

17. [2 marks] Briefly explain what makes **tail resursive functions** much more efficient than non-tail-recursive functions that solve the same problem. Make sure to **explain what is the specification to make a function tail-recursive.**

18. [1 BONUS marks] Make a drawing related to something you learned in this course that you liked.

Mutiple Choice Questions

The diagram below shows a small class hierarchy we may use to build a simulation of cells in an animal's body. Study it carefully and then answer the relevant questions. Class constructors and destructors are NOT SHOWN, but assume they are implemented.



1. The parent class for 'Heart Cell' is:

a) Muscle Cell b) Cell c) Both of a) and b)

d) 'Heart Cell' is the parent class of 'Muscle Cell' e) It's complicated…

2. 'Muscle Cell' inherits the variables 'age' and 'infected' from 'Cell'.

a) TRUE b) FALSE c) It actually contains an entire object of class Cell

3. 'Cell' has a field for 'beats_per_minute' because it was defined by derived class 'Heart Cell'

a) TRUE b) FALSE c) The 'beats_per_minute' only goes up to 'muscle_cell'

4. When we call 'simulate_cell()' in 'Muscle Cell' the following happens:

a) The function 'simulate_cell()' from 'Muscle Cell' is called
b) The function 'simulate_cell()' from 'Cell' is called
c) First the 'simulate_cell()' function from 'Muscle Cell' is called, and then the one from 'Cell' is called
d) We get an error from the compiler because the functions have the same name
e) The compiler pauses to ask the user which function to call

5. When we call 'simulate_cell()' in a "Heart Cell' the following happens:

a) We get a compiler error because there is no 'simulate_cell()' in 'Heart Cell'
b) The 'simulate_cell()' function is 'Cell' is called
c) The 'simulate_cell()' function from 'Muscle Cell' is called
d) First the 'simulate_cell()' from 'Heart Cell' is called, then the one from 'Cell' is called.
e) First the 'simulate_cell()' from 'Cell' is called, then the one from 'Heart Call' is called.

6. If we create a linked list of 'Cell' objects, we can then insert into this list objects from classes 'Cell', 'Muscle Cell', 'Heart Cell', and/or 'Brain Cell'.

a) TRUE
b) FALSE
c) We have to convert them to object of class 'Cell' first
d) It's the wrong thing to do, good software design requires having a linked list for each different class of cells
e) It depends on the compiler version

7. We should really avoid having a mult-level herarchy of classes. instead, we should have all cell types be derved from the single parent class 'Cell'.

a) TRUE b) FALSE c) It's complicated…

8. When should we not make a reasonable assumption? (someone in A48 suggested this one, it wasn't me!)

a) Never (i.e. always make a reasonable assumption!)
b) When we're working in a team and should consult with our team mates
c) Make a reasonable assumption
d) When additional information will result in a more useful solution
e) Always (i.e. never make a reasonable assumption!)

9. Which of the following is something Google API experts **do not say** is a property of a good API?

a) It's easy to extend and improve
b) It's suitable to those who will use it
c) It is implemented in a Object-Oriented Language
d) It is difficult to use incorrectly
e) It does one thing very well

10. Suppose we have BST of integers on which we have carried out a number of **insertions** and **deletions**, if we look at the places different integers occupy in the tree, what can we say about the order in which the values were inserted in the tree?

a) All the integers at leaf nodes were more recently inseted than all non-leaf nodes
b) The most recently inserted integers are visited last when performing in-order traversal
c) Given the integer x at some node, all the integers in ancestor nodes were inserted before x
d) The oldest inserted (the value that was inserted first) is visited first when performing in-order traversal
e) We can ont assert any of the above regarding the order in which integers were inserted

11. The complexity of **removing an edge from a graph with N nodes and |E| edges** when the graph is stored as an **adjaceny list** is:

a) $O(N^2)$ b) O(1) c) O(N) d) O(|E|) e) O(Log N)

12. The complexity of **removing a node from a graph with N nodes and |E| edges** when the graph is stored as an **adjaceny list** is:

a) O(1) b) O(|E|) c) O(N) d) O(Log N) e) O(N Log N)

13. The complexity of a **correctly implemented** related_k_dist() funcrtion (as defined in your A3) for an ingredient graph with N nodes is – *you don't need to remember the function definition, just what it does:*

a) O(k) b) $O(N^2)$ c) O(N) d) O(|E|) e) $O(N^k)$

14. We should avoid using QuickSort() because its worst-case complexity is $O(N^2)$ just like bubblesrot.

a) Always True b) Always False c) False if using a smartly implemented quicksort()
d) True for mission critical applications e) c & d

15. (bonus, no wrong answer here) How was this test?

a) Pretty bad b) I've seen worse c) Fair d) Pretty reasonable e) Really good