速成教育
SPEED UP EDUCATION

CSCA48

Introduction to Computer Science II

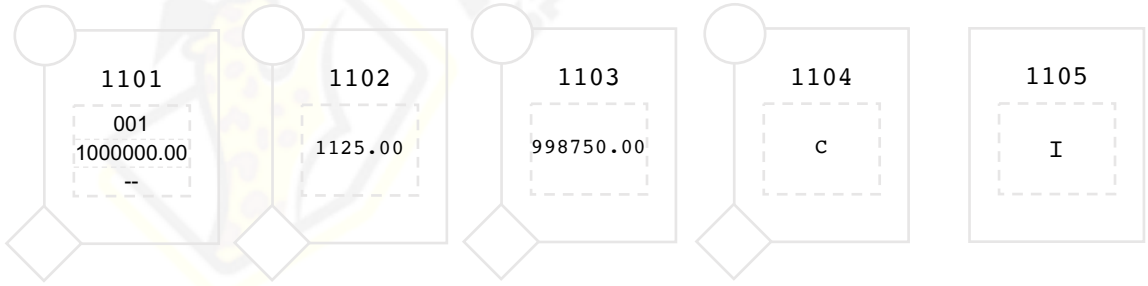
导师： **VC**

2019S CSCA48 Midterm

Unit 1&2 –Memory model, variables, pointers, and arrays

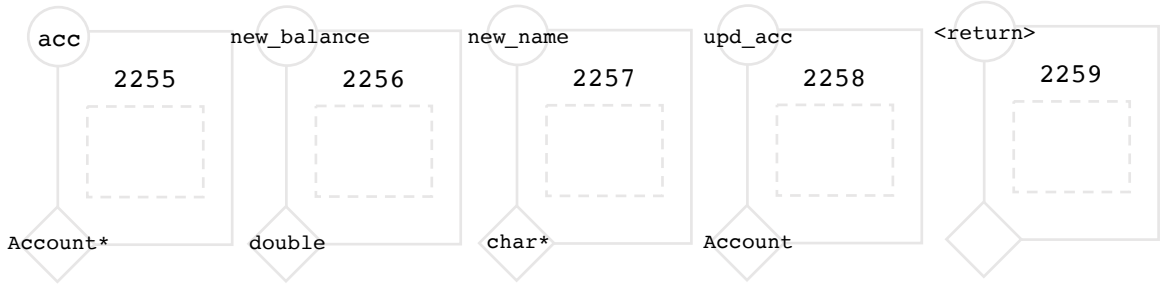
1.- Look carefully at the code below, and answer the questions below.

```
typedef struct bank_account_struct{
    int account_id;
    double balance;
    char branch_name[1024];
} Account;

_______________ updateAccount(Account *acc, double *new_balance, char *new_name)
{
    Account upd_acc;

    strcpy(upd_acc._____________, _____________);

    upd_acc.balance=___________________________;

    return ____________________;
}

int main()
{
    Account my_acc;
    double wd_amount, balance;
    char branch[10]="CIBC";
    wd_amount=1250.00;

    my_acc.account_id=001;
    my_acc.balance=1000000.00;

    balance=_____________________ - wd_amount;
    // Update the branch name with the appropriate call to updateAccount()

    ____________________________________________________________
}
```



Variable for main() go about this line – more boxes are used by branch array, not shown!

Variable for updateAccount() go below this line



1.- [6 marks] Complete the code missing in the program so that it correctly updates the information on my_account

2.- [7 marks] Complete the memory model – variable names and types are missing for main(), and memory contents are missing for updateAccount()

Unit 3 – Organizing, Storing, and Accessing Data

3.- [2 marks] In the space below, provide the definition in C for a compound data type that will be used to store information about super-people (because anyone can be supserhero!)

Super-people have the following attributes:

```
- HeroicName                (which should be cool-sounding, text field)
- RealName                  (a text field)
- Superpower                (a text description of its main ability)
- Awesomeness               (a floating point value in [1, 100.00])
- PagerNumber               (This is also stored as text)
```

```
// Provide here the definition for the compound data type requested above
// The new data type must be named 'SuperPerson'
```

4.- [3 marks] We are building a database of super-people - always useful in case there's a large alien invasion and we need to call them up! Of course, we want to use a linked-list.

In the space below, provide the definition of the data type needed to store ***one node of the linked list.*** The node must contain ***data for one SuperPerson*** (in a field called 'super'), it will also contain a text field indicating the date of their last adventure in a field called `'latest_adventure'`.

```
// Provide here the definition for the compound data type for the list node
// Name this data type 'SuperpersonNode'
```

One of our super-people turns out to be a robot, controlled by a nice A.I. that for whatever reason has decided humans are still smarter, so it carries out whatever assignments are given to it by its human peers, of course, the programming language Super-CI). The robot A.I. keeps its program in a linked list with compond nodes defined as:

```
typedef struct command_node_struct
{
    int command_type;            // ID of the command to be executed
    int command_parameters[5];   // Up to 5 parameter values for this command


    _______________________________ // Sonkething is added here to allow for loops

    struct command_node *next;   // Pointer to next command in the list
} CommandNode;
```

5.- [2 marks] The robot A.I. immediately noticed the structure above does not allow for loops, SO it modified the command_node_struct to allow for the possibility that a command may be a for loop. Show how this is done in the space above.

6.- [2 marks] Is the modification from (Q5) enough to allow the robot A.I. to run nested for loops? In the space below answer ***Yes*** or ***No***, if yes, briefly explain how this would work. If no, show any changes needed for the CommandNode to support nested for loops. Note that we are concerned here *only about the structure of the CommandNode*, not the program that actually runs the command.

Of course it's basically impossible to do anything interesting in programming without if statements, the simplest type of if statement compares a value against zero, and determines what command to execute next depending on whether they are equal or not (later on you will see that such if statements are even implemented at the CPU level in assembly language!)

The following code partially implements the if statement command, it takes as input a pointer to the command node that contains the if statement, a pointer to the head of the linked list of commands, the value to be compared against zero, and the number of the command that should be executed next if the value is not equal to ***zero***. The function returns the pointer to the next command to be executed.

```
CommandNode *ifStatement(CommandNode *currentCommand, CommandNode *head, int value,
                         int next_command)
{
    if (value==0)
    {
        // Should proceed to next command in the command list

        ____________________________________________        // Answer 07 here
    }

    // In this case, the next command to execute is the one whose position in
    // the list is next_command Answer 08 in the space below
```

```
}
```

7.- [1 mark] Complete the part of the function that handles the case where the input value is equal to zero.

8.- [3 marks] Complete the part of the function that handles the case where the input value is not equal to zero.

9.- [10 marks] ***Problem solving***: Suppose we want to take a program written for a robot A.I. that has support for loops, and we want to convert it into a program for a robot A.I. that does not have support for loops. We can do this by unrolling the for loop - that is, replicating the instructions in the loop as many times as needed. e.g.

```
for (i=0; i<3; i++)
{
    printf("Hello!\n");       →after unrolling →          printf("Hello!\n");
}                                                         printf("Hello!\n");
                                                          printf("Hello!\n");
```

Obviously, the for loop may have more than 1 command, so we would have to replicate the entire set of commands in the loop as many times as needed, but we end up with a program with no for loop!

The function below takes an input list of CommandNodes, and returns a new list such that any for loops have been unrolled so there are no for loops left in the resulting list and it can be run a by a robot A.I. with no for loop support.

- ***Assume there are NO NESTED for loops.***
- Assume ***FOR_LOOP_TYPE*** is a constant defined to be the same as the command ID for a for loop.
- Assume you have implemented CommandNode `*insert (CommandNode *head, CommandNode *node)` which insers the 'node' at the tail of linked list with the specified 'head'
- You have a function CommandNode `*copyNode (CommandNode *node)` which makes a copy of the specified 'node' and returns a pointer to it. Q9 4

Complete the function below

```
__________________ unRollLoops (CommandNode *head)
{
    CommandNode *newHead=NULL, *p=NULL;

    while (_____________________________________)
    {

        if (head->command_type != FOR_LOOP_TYPE)
        {
            ____________________________________________

            ____________________________________________

        }
        else
        {









        }

        ____________________________________

    }

    ____________________________________
}
```

10.- [bonus 1 mark] ***DRAW an awesome robot A.I. super-person (ok, super-robotic A.I.)!***

Unit 4 – Complexity

11.- [2 marks] Suppose we have a program for the robot A.I. to run, the program is in a linked list with up to N nodes. It may contain for loops, each for loop could contain up to N commands as well. What is the Big O worst-case complexity of making a copy of the commands in this command list? Briefly explain your answer

12.- [3 marks] ***If the maximum number of iterations for any for loops in the program is N/4***, what is the worst-case complexity (in terms of the number of commands) of executing the commands in the list described in (Q11)?. Justify your answer.

Mutiple Choice Questions

MPORTANT NOTES:

- You can scratch and mark in the questions below ***but please mark just one answer in the bubble-sheet at the back of the test. You will not get marks for multiple scratches.***
- ***Do not forget to record your answers*** - It's strongly recommended you mark your answers as you solve this section to avoid problemns.
- Think carefully - each question is intended to test whether you have fully understood a particular idea or have learned a specific concept from the past 6 weeks.

1.- if we declare an array as ***int my_array[6]***, in memory, we expect to find:

a) One box large enough to contain 6 integer variables, tagged as 'my_array'
b) 6 boxes, each able to hold one int, stored in different places where there is space
c) 6 boxes, each able to hold one int, stored contguously
d) Just a pointer to a memory location that hold the 6 integers
e) One box large enough for 6 int pointers

2.- Which of the following is an ***error in the program*** (the programmer is definitely doing something wrong)

a) Passing an int to a function expecting a double.
b) Passing a char to a function that expects a unsigned char
c) Passing a pointer to int to a function that expects a pointer to char
d) Passing an int to a function expecting a pointer to int
e) All of the options a)-d) are ok, there is no programming error

---

3.- What is the correct way to ***reserve memory on-demand*** for an array of 10 integers?

a) p=calloc(10); b) p=calloc(10,sizeof(int)); c) p=(int *)calloc(10, sizeof(int));

c) p=(int *)calloc(10,sizeof(int)); d) p=10*calloc(sizeof(int));

4.- Suppose that we reserved ***space for an array of 10 integers*** using calloc(), is it possible to use the same memory space to store a string "Hello"?

a) Yes b) No c) It's complicated...

5.- Suppose that we ***reserved space for a string with 10 characters*** and stored "HelloThere" in it. We could expect the following to happen:

a) Program works just fine b) The program produces random output

c) The program crashes (it's killed by the computer) d) Any of a)-c) e) None of a)-c)

6.- Which of the options below is something ***we can not do*** with compound data type variables?

a) Create an array with many of them b) Pass them as parameters to functions
c) Create them on demand using calloc() d) Store them inside other CDTs
e) Compare them to one another

7.- A linked list node ***has to be a compound data type***.

a) TRUE b) FALSE c) It's complicated...

8.- One of your colleagues implemented a little database for keeping track of the food inventory at a supermarket. Your colleague thought it would be a great idea to use binary search, and implemented functions to access entries in the sorted linked list by index.

What is the ***worst case*** complexity of the function that accesses an arbitrary entry in the linked list? Assume Log is the logarithm base 2.

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

9.- ***Think carefully:*** What is the complexity of doing binary search on a linked list using the function that accesses an entry in the list by index?

a) $O(N^2)$ b) O(Log(N)) c) $O(N^3)$ d) O(N) e) O(N Log(N))

10.- What is the complexity of finding ***duplicate entries*** in ***a sorted array of integers?***

a) O(Log N) b) O(N) c) $O(N^2)$ d) O(N Log(N)) e) $O(N^3)$

11.- Because of complexity considerations, linked-lists are not recommended for ***any*** applications that require managing large datasets

a) TRUE b) FALSE c) It's complicated...

That's it for the midterm!