CSCA48

Introduction to Computer Science II

导师： VC

2024W CSCA48 Midterm

S P E E D U P E D U C A T I O N

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 (after the program is done with line 23 and before line 24 starts)***.
Assume the variables are assigned to boxes ***in the order they declared and each box has sufficient space for the variable.***

```
1  int fd(int *array, int length) {
2     int i;                        // Use memory starting at 801 for fd()
3     for (i = 1; i < length; i++)
4        *(array + i - 1) = *(array + i) - *(array + i - 1);
5     return (i - 1);
6  }
7  int main() {
8     int n1;                       // Use memory starting at 0791 for main()
9     int arr[4] = {5, 7, 10, 11}; // Use the back of the page for scratch work
10    int n2;                       // Record only below your final answer.
11    int *p = NULL;                // If we can't read your answer, you won't get marks.
12
13    p = arr;
14    n1 = fd(p, 4);
15    p = arr + 1;
16    *p = 10;
17    *(p - 2) = -1;
18    p = p + 1;
19    *(--p) = 6;
20    *(--p) = 8;                   // ← Question 2 is about this line
21    n2 = fd(p + 1, n1 + 1);       // ← Question 3 is about this line
22    p = p + 4;                    // ← Question 4 is about this line
23    p[0] = 100;
24    return 0;
25 }
```

0791	0792	0793	0794	0795
0796	0797	0798	0799	0800
0801	0802	0803	0804	0805

2.- [2 marks] True of False – when the computer runs **line 20** in the program from the previous page, the program may crash because it's accessing invalid memory locations. **Briefly explain your reasoning.**

3.- [2 marks] True or False – when the computer runs **line 21** in the program from the previous page, the program may crash because it's accessing invalid memory locations. **Briefly explain your reasoning.**

4.- [2 marks] True or False – when the computer runs **line 22** in the program from the previous page, the program may crash because it's accessing invalid memory locations. **Briefly explain your reasoning**.

5.- [2 marks] True or False – we want to keep the implementation of the fd function unchanged, but have different function prototypes. The following two are both valid (**Briefly explain your answer**).

```
1) int fd(int p[], int length);          2) int fd(int p[5], int length);
```

Unit 3 – Organizing, Storing, and Accessing Data

The Toronto central island daily bicycle rental business is getting busier and
they need your help to keep track of all the bikes.
To help with this let's put together a DB to keep track of bikes.
Here's a CDT we can use to store bike information.

```
typedef struct bike_struct
{
     int id          // each bike has a unique identifier
     int type;       // =0 single, =1 tandem, =2 2-seater, =3 4-seater
     int out;        // default*= -1, out time in minutes, 8:15 => 495 (60*8+15),
     int in          // default = -1, return time in minutes
     int rate;       // The rental rate in cents per hour
     char renter_id[256];      // default = "", government issued ID of the renter,
} Bike;
```

We will create two **linked lists**. One is for storing all rented bikes and one is for storing all free bikes that are not rented out yet. The CDT we will use for this is a standard linked list node:

```
typedef struct bikeDB_node
{
    Bike bike;                    // The bike
    struct bikeDB_node *next;     // Pointers to the next node
} BikeNode;
```

6.- [10 marks] Complete the code below. The code should **allocate and initialize** a new node to be added to the linked list for free bikes, with a bike that contains the information provided in the call to the function. The initial values for the fields not listed in the parameter list of the function are set to **default** values according to the comments in the struct `bike_struct` definition.

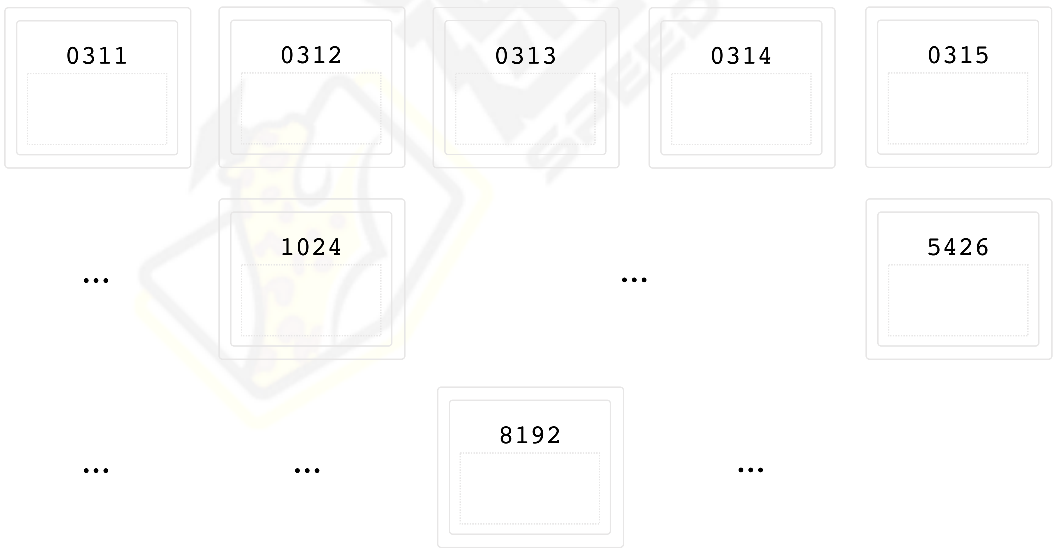
```
___________________ newBikeNode (int id, int type, int rate) {

// Allocates and initializes a new node so we can add it to the linked list.
// The information for the bike is passed into this function as input arguments.















}
```

Suppose that we have implemented linked list insert, and that
our insert function **adds new bikes at the head of the list** which keeps track of free bikes.
Now suppose we use it in main() to add a couple bikes to the available bikes DB, as shown below

```
int main()
{

     BikeNode *p_bike;

     BikeNode *head_free = NULL;

     p_bike = newBikeNode(123,1,1000);    // Gets locker #8192

     head_free = insertBike(p_bike);

     p_bike = newBikeNode(2024,2,2000);   // Gets locker #5426

     head_free = insertBike(p_bike);

     p_bike = newBikeNode(3210,4, 3800); // Gets locker #1024

     // Show what the memory model looks like at this point!
}
```

7.- [6 marks] Show what the memory models would look like after we have run the code in `main()` above.
Include the memory for main (), as well as anything that has been allocated for the linked list.
For `main()`, use lockers starting at 0311. You can use the back of the page for scratch work, **record only your final answers here.** If we can't read it, or it's not clear, we can't mark it.

***** Update the pointers in BikeNode, and use arrows to show clearly which node points to which so we can see the structure of the linked list - this also goes for any BikeNode in*** `main()` ******



Unit 4 – Complexity of list operations

8.-[4 marks] We need to a write a function to search the bikes DB to find the bike when a customer returns it. Show in **very clear pseudocode** how this function should be implemented, give the Big-O estimate of its complexity, and briefly justify your answer

9.- [5 marks] We want to write a function to handle bike return. The function returns the rental charge in cents, move **the returned bike from the rented bike list to the free bike list** and updates the data item fields accordingly. Give **very clear pseudocode** showing how the function works, give an estimate Big-O complexity, and briefly justify your answer.

Mutiple Choice Questions

For questions 1 to 5 consider the snipped of code shown below:

```
typedef struct patron_struct {
     char name[512];               // Name of the Patron
     int num_books;                // Number of books a patron can borrow (in 2-10)
     char book_list[10] [1024];    // An array with 10 book titles, each title has
                                   // is a char array with length of 1024.
} Patron;

int main() {
     Patron patron1, patron2;
     Patron *p=NULL;
     p=&patron1; W
     // Answer questions 1 to 5 based on the code
}
```

1.- What's the correct way to set the number of books to 5 ***using the pointer 'p'****?*

a) p→patron1.num_boos = 5; b) p→*num_books = 5; c) *(p→num_books) = 5;

d) (*p).num_books = 5; e) p→num_books = 5; f) &(p.num_books) = 5;

2.- How would we set the patron's name to 'John Doe' using the "patron1" ***variable****?*

a) patron1.name = "John Doe";

b) *(patron1.name) = "John Doe";

c) strcpy(&patron1.name, "John Doe");

d) strcpy(patron1.name, "John Doe");

e) strcpy(*(patron1.name), "John Doe");

3.- How can we set the title of the 3rd item in the patron to "GDB Cookbook" using the ***pointer 'p'***? Note that the → operator has higher precedence than the & operator.

a) strcpy(*(p → book_list[2]), "GDB Cookbook");

b) p → book_list[2][] = "GDB Cookbook";

c) *(p → book_list[2]) = "GDB Cookbook";

d) strcpy(&p → book_list[2][0], "GDB Cookbook");

e) strcpy(&p → book_list[3][0], "GDB Cookbook");

4.- How would we ***copy the information in "patron1"*** so that "patron2" has the same exact content, ***using the pointer "p"?***

a) patron2 = p → patron1;   b) patron2 = &p;   c) *(patron2) = *p;

d) *patron2 = &patron1;   e) patron2 = *p;

5.- How many boxes are reserved in memory for main()?

a) 3   b) 4   c) 5   d) 2*(1 + 512 + (10 * 1024)) + 1 + 1   e) 8

6.- Suppose we're using an ***int variable 'd'***, and we want to print it using printf(). however, we call printf like so: ***printf("The value of d is %f\n", d);*** What do we expect will happen? Assume the gcc/clang compiler only uses the -Wall flag to compile the code.

a) The compiler will throw an error because the types don't match, and it will ***not compile*** the code.

b) The variable ***'d'*** is converted to a ***floating point type,*** and then the ***integer part of d is printed followed by a couple of zeros after the decimal point.***

c) The program compiles but ***crashes (fails)*** when we run it.

d) It prints a floating point value, where the ***value is incorrect.***

e) The compiler automatically corrects ***%f*** to ***%d*** and prints the value of ***d*** properly.

7.- Which of the options below correspond to ***something we cannot do with compound data types variables?***

a) Copy the contents of a CDT variable onto another of the same CDT type.

b) Use a pointer to access different fields in the CDT variable.

c) Compare two CDT variables of the same type (e.g., to sort them in some order).

d) Return a CDT variable from a function.

e) Create arrays of CDT variables.

8.- Which of the statements below about linked lists ***is NOT true?***

a) They allow us to store only the items that are currently in the collection. Space is never pre-allocated or wasted.

b) They support an arbitrary number of items (as long as there is memory), and capacity is not fixed.

c) Similar to arrays, they support random access of an item at a position in the list in constant time.

d) They are the simplest data structure that can maintain a changing-size collection with little or no space overhead.

e) They can be used to store more diverse and complex information than arrays.

____

9.- When your code does not work as intended, which is the best strategy that you should follow in order to fix the problem?

a) Ask your TAs or instructors during an office hour why your code is not working.

b) Test and debug your code, then find an instructor during office hours if you are not sure how to solve the problem.

c) Send your code to your instructor(s)/TA so they can fix it for you.

d) Re-run your code several times; eventually, it may work as expected.

e) Ask someone on Piazza what to do.

____

10.- Which of the following statements about programming in C ***is NOT true?***

a) All variables must have a data type, and this data type is fixed and can be changed later by type conversion.

b) Function arguments can have any type, and we can have as many as we want, in any order.

c) For loops can use counter variables with any simple (not array, not CDT) data type.

d) We can have pointers for any variable, array, or CDT variable.

e) If we pass an array to a function, the function knows the size of the array.

____

11.- Which of the statements below corresponds to something ***the compiler allows with arrays?***

a) Assign a pointer to the array name.

b) Change the size of the array at runtime.

c) Pass arrays as arguments to functions so they can work with their values.

d) Access data that is beyond the size of the array.

e) Change the data type of the array elements.