

CS1020E: DATA STRUCTURES AND ALGORITHMS I

Tutorial 1 – Basic C++, OOP Problem Solving


(Week 3, starting 22 August 2016)

1. Evaluation Order (Note: You can use any other C++ code editor/compiler).

Examine the code snippet. What is the **output**, and **why**?

Tip: Check your answer! Create a program in vim, paste main method within. Compile and run in sunfire.

```
int main() {
    int a = -1, b = 1, c = 1, d = 0, e = 2, f = 2, g = 0;
    int h = f-- && e++ && d++ && c-- || b++ || a++;
    if (g = 9) {
        cout << a << b << c << d << e << f << g << h << endl;
    } else {
        cout << h << g << f << e << d << c << b << a << endl;
    }
    return 0;
}
```



a	b	c	d	e	f	g	h

2. Understanding Pointers

In OOP languages, pointers (so named in C++; may be called “references” in other languages) are widely used to locate one object from another. It is necessary to have a firm understanding of them.

For each of these cases, *independent of one another*, **draw** how the variables may appear in memory, and what **output** you would expect. You do not need to worry about the exact memory addresses, but you should find out how different expressions are related to each other. ☺, ☹, ○, and ◆ represent memory addresses.

(a) has been done for you. (d) to (g) may be more difficult. Remember to check your answer.

(a) <pre>int i = 3; cout << &i;</pre>	(b) <pre>int* p = new int(3); cout << &p << p << *p;</pre>	(c) <pre>int* ap = new int[3]; for (int i = 0; i < 3; i++) ap[i] = i - 1; cout << &ap << ap << *ap << ap[0];</pre>
(d) <pre>int i = 3; cout << *&i;</pre>	(e) <pre>int* p = new int(3); cout << *&p << *&p << **&p;</pre>	

```

(g)
int* p = new int(3);
int** dp = &p;
int*** tp = &dp;

cout << *tp << &***tp <<
      &***tp << **&tp;

(f)
int** dp = new int* [3];

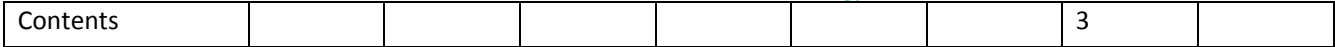
for (int i = 0; i < 3; i++)
    dp[i] = new int(i-1);

cout << &dp << dp << *dp <<
      dp[0] << **dp << *dp[0];

```

(a)

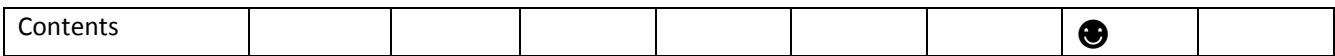
Address



The output &i is 😊.

(b)

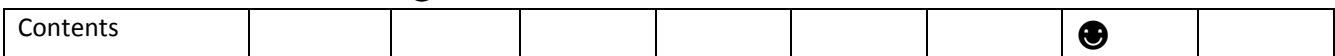
Address



The output is

(c)

Address



The output is

(d)

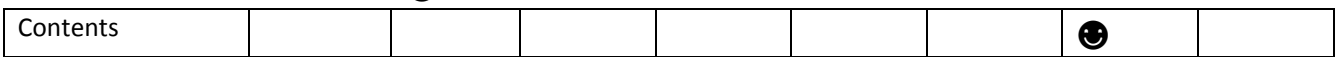
Address



The output is

(e)

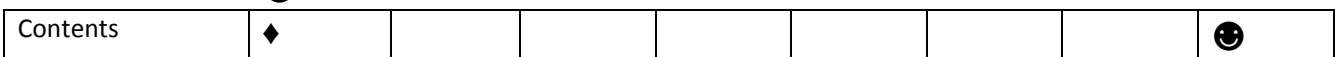
Address



The output is

(f)

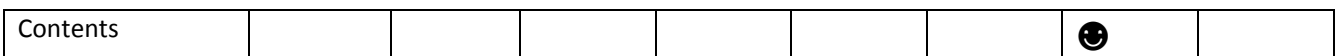
Address



The output is

(g)

Address



The output is

Also, when the **new** keyword is used, the system dynamically allocates memory for the newly created object. Therefore, the object ends up in a portion of memory called the **heap**. Otherwise, when **new** is not used, objects assigned to variables declared within functions reside on the **stack**.

Why do we need to bother about heap vs stack memory? What other keyword and syntax is/are involved?

3. Object-Oriented Programming

You want to print out the lyrics of this song¹, to teach (or confuse) kids about the sounds animals make:

```
Dog goes woof
Cat goes meow
Bird goes tweet
Mouse goes squeak
Cow goes moo
```

The lyrics can be generalized for different animals, each having a different *name* and **sound**. With knowledge of object-oriented programming, you want to demonstrate that it is possible to write a program that *displays* the song. To show that your program works, add the 5 animals above and test your program.

Don't forget to include the necessary *system header*, and use the appropriate *namespace*. Use the skeleton on the next page to solve the problem:

```
class Animal {
    /* TODO: Implement data and functionality of an Animal here */
};

class Song {
private:
    Animal** _animals;
    const int _size;
public:
    Song() { /* TODO: Create your zoo, an Animal* array */ }
    ~Song() { /* TODO: Cleanup the 5 animals and the array... */ }

    void display() {
        for (int i = 0; i < _size; i++)
            cout << endl; /* TODO: Add the lyrics here... */
    }
};

int main() {
    Song song;
    song.display();
    return 0;
}
```

- Hope you had fun, prepare well for tutorial 2 ☺ -

Draw diagrams
Attempt tutorials
Test your solution

¹ Adapted from "The Fox" by Ylvis, 2013