CS 7B Midterm 1–Solutions - Fall 2017 - 11/1/17

1. Write the number of the definition on the right next to the term it defines.

(a) **class** 3

(b) **constructor** 4

(c) **container** 12

(d) **copy** 1

(e) **invariant** 10

(f) **overload** 7

(g) **reference** 6

(h) **pointer** 5

(i) **scope** 2

(j) **byte** 11

(k) **type** 9

(l) **move** 8

(1) An operation that makes two objects have values that compare equal..

(2) The region of program text (source code) in which a name can be referred to.

(3) A user-defined type that may contain data members, function members, and member types.

(4) An operation that initializes an object. Typically establishes an invariant and often acquires resources needed for an object to be used (which are then typically released by a destructor).

(5) (1) A value used to identify a typed object in memory; (2) a variable holding such a value.

(6) (1) A value describing the location of a typed value in memory; (2) a variable holding such a value.

(7) Define two functions or operators with the same name but different argument (operand) types.

(8) An operation that transfers a value from one object to another, leaving behind a value representing "empty."

(9) Something that defines a set of possible values and a set of operations for an object.

(10) Something that must be always true at a given point (or points) of a program; typically used to describe the state (set of values) of an object or the state of a loop before entry into the repeated statement.

(11) The basic unit of addressing in most computers.

(12) An object that holds elements (other objects).

2. Consider the following complete program:

```
1 int main() {
      char* pc = 0;
3     char& rc = *pc;
      std::cout<<pc;
5     std::cout<<rc;
  }
```

(a) What is the variable name declared on line 3 and what is its type?
ANS: The variable named `pc` is declared to be a pointer to type `char`.

(b) How is the variable on line 3 initialized?
ANS: It is assigned the `NULL` pointer (`nullptr`)

(c) What is the variable name declared on line 4 and what is its type?
ANS: The variable named `rc` is defined to be a reference to (the address of) a variable of type `char` whose value is the address pointed to by `pc`.

(d) How is the variable on line 4 initialized?
ANS: It is assigned the value of the address of the `char` pointed to by `pc`, which is then an alias for the `char` that `pc` purportedly points to...except `pc` is `NULL`. Bad!

(e) This code will likely compile but then crash on execution. Why?
ANS: Line 4 binds the reference `rc` to `nllptr`. As the C++11 ISO states, `[dcl.ref] [...] a null reference cannot exist in a well-defined program, because the only way to create such a reference would be to bind it to the "object" obtained by dereferencing a null pointer, which causes undefined behavior.`

3. Consider the following code:

```
#include<string>
int main() {
    std::string& rs;
}
```

Why does it not compile?
ANS: error: 'rs' declared as reference but not initialized.
By contrast, this code (below) will compile since the reference is initialized to something other than nullptr:

```
#include<string>
int main() {
    string s("xyzzy");
    string& rs = s;
}
```

4. Consider the following complete program:

```
#include<iostream>
int main() {
    char* ps = new char;
    std::cout << ps;
}
```

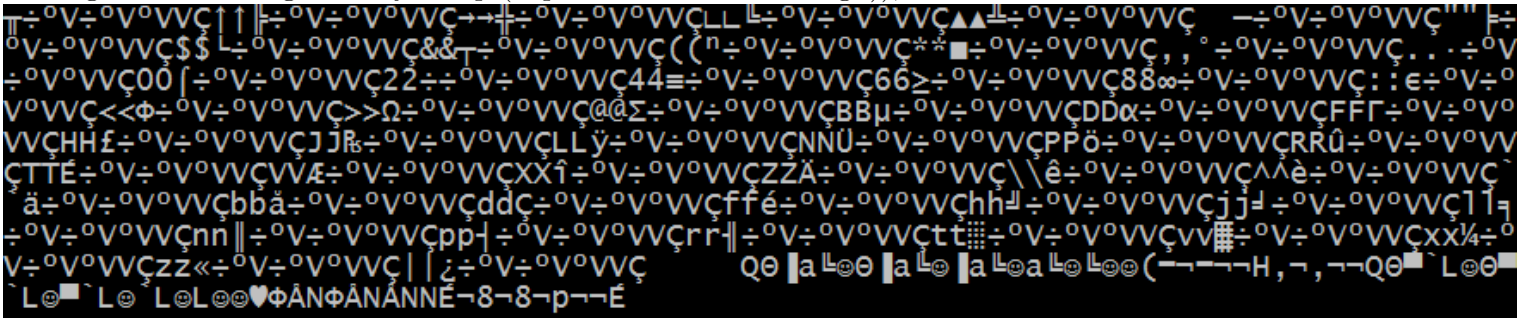
Would you expect this program to compile and run ok? If so, what sort of output might you expect and why?
ANS: It would compile and print some random character, "a" or similar. This is because of the special nature of how C-strings are handled by the compiler. If, by contrast, you had

```
#include<iostream>
int main() {
    int* pi = new int;
    std::cout << pi;
}
```

In that case you would get the memory address that the pointer points to. Try a little experiment:

```
#include<iostream>
int main() {
    char* pc = new char;
    while(pc--)
        std::cout << pc;
}
```

I get the following memory dump (hope it's not incriminating:-)), which then crashes:



5. Consider the following complete program:

```
1 #include<iostream>
  int main() {
3     char* ps;
      std::cout << ps;
5 }
```

Would you expect this program to compile and run ok? If so, what sort of output might you expect and why?
ANS: It would likely crash: uninitialized, unallocated memory is being attempted access.

6. Consider the following complete program:

```
1 #include<iostream>
  #include<string>
3 int main() {
      std::string* ps;
5     std::cout << ps;
  }
```

What sort of output might you expect and why?
ANS: Expect a memory address allocated by the compiler (like `0x40196e`).

7. Consider the following complete program:

```
#include<iostream>
using namespace std;
void to_lower(char* s) {
    while(*s!=0)  /// or, while(*s!='\0'), or while(*s!=0)   {
        if(*s >= 'A' && *s <= 'Z') ///if upper case
            *s += 32; /// convert to lower
        ++s;
    }
}
int main()
{
    char arr[] = "Halo, World!";
    char* str = arr; /// arr == &arr[0]
    /// this replaces
    /// char* str = "Halo, World!"
    cout << str << endl;
    to_lower(str);
    cout << str;
}
```

(a) What is the loop condition of the `while` loop in `to_lower()`?
ANS: The boolean is `*s!=0`

(b) Could this condition be changed to simply `*s` while preserving the meaning of the code? Why or why not?
ANS: Yes, since the condition would still be that `*s` is not the `NUL` character.

(c) Could this be changed to `*s!='0'` and preserve the meaning of the code? Why or why not?
ANS: No. This change would result in the strings "`SOOP`" and "`SOUP`" being handled differently.

8. Write a function `reverse()` that will take a c-string (`char*` type) input and copy the c-string in reverse order into memory it allocates on the free store. Do not use any standard library functions. Do not use subscripting; use the dereference operator * instead.
ANS: The code at right does the trick. The output is

```
Halo, World!
!dlroW ,olaH
```

Note that the local copy of the `char*` would be deleted when it went out of scope, but it's good practice to delete anyway.

```
#include<iostream>
using namespace std;
char* reverse(char* scopy) {
    int cnt{0};
    while(*scopy!=0)  {
        ++scopy;
        ++cnt;
    }
    char* rev = new char[cnt];
    for(int i = 0; i < cnt; ++i) {
        --scopy;
        rev[i] = *scopy;
    }
    delete [] scopy;
    return rev;
}

int main() {
    char arr[] = "Halo, World!";
    char* str = arr;
    cout << str << endl;
    str = reverse(str);
    cout << str;
}
```