Bit manipulation: shifting and masking
What: operations on ints/longs/chars that perform bit-by-bit comparisons, result based on the operation performed.
Where: C, C++, Java

Masking: bit-by-bit comparison, only a portion of the source shows through.
NOT ~ ~0 => 1, ~1 => 0

Bits in action:
src: 1010 1010
~src: ________

Example in code:
dst = ____

Intuition:
Use: __________________________________________

AND: & 1&1 => 1, else => 0

Bits in action:
src: 1010 1010
&mask: 1111 0000
dst: _______________

Example in code:
dst = ________________;

Intuition: ANDing w/ 1s => ______, ANDing w/0s => ______
Use: ________________, ________________

Practical Use:
Extract one hexadecimal digit from a number in a hexout method:
digit = number & 0x????????;  // extracts last digit

number: 0x12345678
&mask: 0x????????
digit: _________________ => digit extracted
OR

|      | 0|0 => 0, else 1 |

Bits in action:

- **scr:** 1010 1010
- **|mask:** 1111 0000
- **dst:** __________

Intuition: ORing w/ 1s => ________, ORing w/0s => ________

Use: ________________

Practical Use:

Convert a decimal digit to ASCII:

```plaintext
code = digit | ‘0’;  // 7 | 0x30 => 0x37…”7”
```

- **scr:** 0000 0111
- **|mask:** ?????????
- **dst:** ________________ => ASCII code for ‘7’

XOR

^ 0^1 => 1, 1^0=> 1, else 0

Bits in action:

- **scr:** 1010 1010
- **^mask:** 1111 0000
- **dst:** __________

Intuition: XORing w/ 1s => ________, XORing w/0s => ________

Practical Use:

- ________________
- ________________
- puzzle: xxx ^= yyy; yyy ^= xxx; xxx ^= yyy;
Shifting: Source value shifting by an amount in bits
   >>  right shift
   <<  left shift

Ex:  3:  _______________  // 3 shifted by ___ bit to the ___
     6:  _______________  // 3 shifted by ___ bit to the ___
     12: ________________ // 3 shifted by ___ to the ___
     24: ________________ // 3 shifted by ___ to the ___
     48: ________________ // 3 shifted by ___ to the ___

Shifting left _______________ by powers of ___,
Shifting right _______________ by powers of ___

Ex: in decimal: 123 x 100 = 12,300

Ternary operator: if-then-else expression

(condition) ? true_value : false value

Ex:  
   aaa = (bbb == 10) ? 20 : 30;

// same as:
   if ( )
      aaa = ;
   else
      aaa = ;

Reversing stack1 onto stack2:

while (__________) {
   __________;
   __________;
   __________;
   __________;
}
How to declare and initialize an array:
int array[] = {1,2,3,4,5,6,7};

Parallel arrays:
What: Two (or more) arrays that share the same ________:
For us: __________, __________
_____________________________________________________

Data hiding in C:
In your .h files (Header files) is where you list __________ information.
In your .c files (Code) is where you list __________ or __________ information.

What goes in a .h file:

Ans:
_____________________________________________________

Reasons:
1. ____________________________________________________________________________
2. ____________________________________________________________________________
static: prefix qualifier to an identifier that implies:
- ________ to a file, method, class
- ________ section allocation

variables:
- static variable declared outside of a function:
  - Ex: static int xxx;
    o data section, initialization to 0, created when program runs

- static variable declared inside of a function:
  - Ex: func() { static int xxx; ... }
    o data section, initialization to 0, created when program runs
    o location known by the variable name only when
      ________________
    o local in scope to __________
    o effect: __________ its value between calls
    o __________ instance of that variable

- static variable declared inside a class:
  - Ex: class XXX { public static int xxx;...}
    o data section, initialization to 0, created when program runs
    o scope is local to __________
    o __________ part of the “sizeof” an object
    o __________ in the same space in memory as objects of class
    o __________ instance of that variable even if
      ____________________________

static method not in a class: local to __________, callable by a function that ______________________________.

static method in a class:
- method callable without __________________________
- benefits of __________________________
  (____________________________________)
“struct”:
What: A keyword in C/C++ that combines multiple variables/fields into one unit.
- like a “class”, no classes exist in C. Classes are only in Java and C++.
- in C, struct _________ automatically produce a user-defined type.
- in C, no __________, no __________, no __________, no __________.
- in C, a struct are fields are ________.
- in C++, struct is almost the same as a class
  o C++ has constructors, destructors, accessors, access rights
  o A “struct” in C++ has all fields ________ by default.
  o A “class” in C++ has all fields ________ by default.

Why:
- allows us to create __________ in C.

Syntax:
struct tag {  // tag sometimes needed, sometimes isn’t
    int xxx;  // field 1
    int yyy;  // field 2
};

Combine “structs” with “typedef” to create a object definition:

Syntax of typedef: typedef existing_type new_type;

typedef struct Student {  // tag is Student
    char name[20];
    int number;
} Student;

// existing type is in blue
// new type is in green

typedef struct {  // no tag
    char name [20];
    int number;
} StudentWithTag;

typedef struct StudentWithTag {
    char name [20];
    int number;
    struct StudentWithTag * sibling;
} StudentWithTag;

// resulting type: “________________________”
struct StudentWithTagNoTypedef {  // typedef is absent
    char name [20];
    int number;
};
main () {
    Student stu1;  // RTS object: size is ___________ bytes
    StudentWithTag stu2;  // RTS object
    StudentWithNoTag stu3;  // RTS object

    // struct keyword is mandatory at declaration
    struct StudentWithNoTagDef stu4;

    // RTS pointers that can point to ______________
    Student * stup1, *stup2;

    stu1.number = 123;  // to assign to a field given an object

    // stup1 points to the _______________
    stup1 = (Student *) malloc (sizeof (Student));

    // stup2 points to the RTS
    stup2 = &stu1;

    // to assign to a field given a pointer
    stup1->number = 123;
    stup2->number = 234;

    free (stup1);  // avoid memory leak
}  // all ________ objects are deallocated here

Syntax:
    object . field  // to access a data field of an object
    pointer -> field  // to access a data field of an object
    (*pointer).field  // syntactically correct, but ______
    (&object)->field  // use identifies author as a ______

Design decisions:

Tiers or levels or layers of code:
- Each tier or level is distinct.
- Each tier related to surrounding tiers through calling methods in _______ tiers, being called by methods of _______ tiers.
- Each method needs to work based on its _______ and _______ values.
- When all the tiers are put together, you have _______.
- Like a layer cake.

Method of which object should change data fields of that object?

___________ is responsible for __________________________
Compiler Behavior
- What is a compiler?
  - Ans:
    o A programming tool that _____________________.
    o For us: C, Java, C++, to eventually get byte code, native machine code.

- What do we want a compiler to do for us?
  - Ans:
    o ____________________
    o ____________________

Compiler errors:
1) Lexical errors: Input is not recognized as legal in terms of basic symbols.
   - First errors the compiler identifies
     a. ________________________
     b. ________________________
     c. ________________________
     d. ________________________
     e. English: __________________

2) Parse errors or Syntax errors: Group of tokens is legal, but the order is unexpected.
   a. ________________________
   b. English: __________________

3) Semantic error: Structure is legal, but your statement doesn’t make sense.
   a. ________________________
   b. ________________________
   c. ________________________
   d. English: __________________
4) Warning: The compiler will do as you ask, but not necessarily as you want.
- in CSE 12, we will compile without warnings
  a. ______________________
  b. ______________________
  c. ______________________
  d. ______________________
e. English: ______________________

5) Advice: All errors beyond the first are based on the compiler making a guess at what you meant:
  a. Guess was good: Error message is _____.
  b. Guess was wrong: Error message is ______.
  c. Therefore…correct the first few messages and compile again
  d. Therefore…don’t even look at 20th message

Run-Time Errors:
  What: Your program asked to do the impossible.
  Ex: if (10/0)…
- access memory that is out of range.
- result: ________ in C/C++, __________ in Java
- The “core” file:
  o A run-time image of your program in execution just before you asked to do the impossible.
  o Often very large files (consume entire quota)
  o What good is it?
    • The ____________ can read it
      o gdb a.out core
      o where … tell the line of problem.

Side note:
Desire to remove core file, xxx.o: student types (note the extra unwanted space after the * before the .o):
  rm core * .o
* is expanded by UNIX to match all files in current directory, and they will all be removed.
  make clean    // safely remove core, exe, .class, .o
  make backup   // copies source into a backup directory
Logic Errors: The program does what you ask, not what you want.
- Most of your time will be spent here.
  Ex: Output is wrong.
  Ex: infinite loop.
  Ex: extra output of display.

Driver program:
What: Unit testing: Test each function/method individually

  hw1-2: driver1, driver2
  hw3: driver
  hw4: no driver (executable is calc, an end user application)
  hw5: driver1, driver2
  hw6-7: Driver1, Driver2
  hw8-9: Driver

Who is the end user of the driver/Driver programs?

Ans: ________________

Therefore: All input doesn’t need to be checked, because ________________.
Purpose: To test your methods before ______________________.

Therefore: The driver is disposable…you can throw it away once you are done testing (not in CSE 12).