Exam1 Review Topics

Data Structures

You must justify all your answers to recieve full credit

Midterm 1 has 4 theory questions and 1 short C++ program. Total time is 90 minutes.

1. **C++ apart from Object Orientation**
   (a) Restore/drop parentheses in expressions given precedence and associativity rules.
   (b) Translate between flowcharts and C++ control structures.
   (c) Use bit arithmetic (bitwise AND, OR, NOT, XOR, left/right shifts).
   (d) Analyze side-effects in operations, including arrays, increments, short-circuit Boolean evaluation.
   (e) Run pseudocode pointer operations on struct datatypes, draw arrows.
   (f) (C++) Perform bit manipulation (rotation etc.) on a C++ fundamental type.
   (g) (C++) Manipulate arrays, including char arrays (C-strings), also 2D arrays.
   (h) (C++) Input data using `sstream`, `getLine()`, `get()`, `peek()`.
2. **C++ with Object Orientation**
   (a) Parameter passing to functions by value/reference, `const` modifier, default parameter values.
   (b) How class inheritance works on public, protected, private fields.
   (c) The order how constructors and destructors are executed.
   (d) (C++) Given the usage pattern write function prototype (param types and modes).
   (e) (C++) Implement inheritance with virtual/non-virtual functions.
   (f) (C++) Custom comparison function to generic sorting, max or similar algorithm.
3. **Big-O, Omega, Theta Notation**
   (a) Find the asymptotic growth for a given function.
   (b) Compare classes of function growth or order them.
   (c) Express time complexity for recursively defined functions.
   (d) Express time complexity for a code snippet estimating "from the inside out".
   (e) Find the expected time complexity given input probabilities.
   (f) Find the amortized time complexity for an operation on a given data structure.
   (g) (C++) Count the number of calls for certain methods only (e.g. comparisons).
4. **Lists, stacks, queues.**
   (a) Describe a list as an Abstract Data Type (ADT).
   (b) Create an array implementation of a list.
   (c) Create a singly linked list implementation of a list.
   (d) Create a doubly linked list implementation of a list.
   (e) Use a stack-based algorithm and visualize the content of the stack.
   (f) (C++) Use STL classes (`std::list`, etc.) with their iterators.
   (g) (C++) Implement some ADT operation (on list, stack, queue) using pointers.
   (h) (C++) Implement some ADT operation (on list, stack, queue) using arrays.

1 C++ apart from Object Orientation

1.(a). Restore/drop parentheses in expressions given precedence and associativity rules; convert abstract syntax trees into expressions (or expressions back into syntax trees).

(a) Write C++ code that implements the syntax tree shown in the Figure 1.

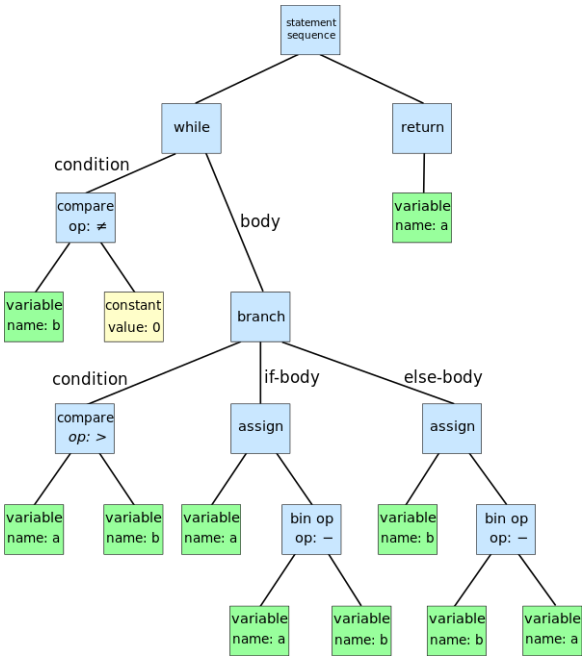

Figure 1: Syntax Tree

(b) Draw the abstract syntax tree for this expression:

```
cout << ( a && b + c ? 3 : 5 - d / e ++ / f ) << ( g = h << 17
== - i > 15) << endl;
```

In syntax trees leaves are variables such as `cout`, `a`, `b` or numbers. Operations such as shifts, Boolean and regular arithmetic are internal nodes.

(c) Restore all parentheses in the expression:

```
cout << ( a && b + c ? 3 : 5 - d / e ++ / f ) << ( g = h << 17
== - i > 15) << endl;
```

1.(b). Translate between flowcharts and C++ control structures:

(a) Draw the flowchart that corresponds to this code snippet:

```
for (int i = 0; i < 3; ++i) {
  switch (a) {
    case 1: if (b > 0) break;
    case 2: a += 1;
    case 3: b = a - 17; break;
    default: b = a - 17; continue;
  }
}
```

Please note that every block in the flowchart belongs to one of the 5 kinds: a unique oval-shaped "Start" node, a unique oval-shaped "End" node, rectangular-shaped node with an atomic statement (such as assignment or increment), diamond-shaped branch node which checks a condition and has two branches for `true` and `false`, and finally - a shaded circle-shaped node where two branches are joined.

(b) Convert the flowchart in Figure 2 to C++ code:

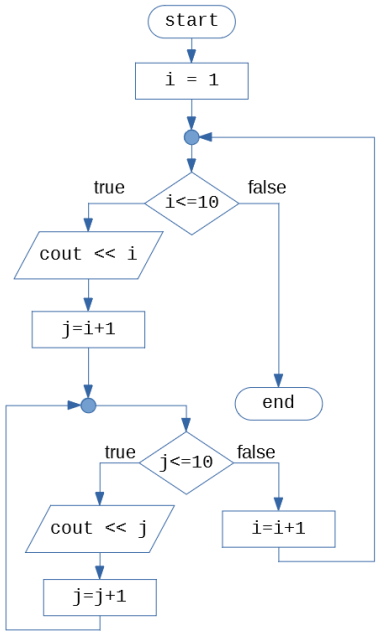

Figure 2: Syntax Tree

(c) Convert the following code to a flowchart:

```
bool a,b;
if (a)
  if (b) { f(); }
else { g(); }
```

Is it possible to insert a few curly braces in order to alter the flowchart? How can it be done? Write that new version of C++ code.

1.(c). Use bit arithmetic:

(a) Consider the following C++ code fragment computing Bitwise XOR:

```
int a = 0x00abcdef;
int x; cin >> x; //
int b = a^x;
cout << hex << b;
```

It is known that the expression prints `01654321`. Find the value of `x` that was received from the input.

(b) Consider the following C++ code fragment computing Bitwise AND:

```
int a = 0x00FF00FF;
int x; cin >> x; //
int b = a & x;
cout << hex << b;
```

It is known that the expression prints `00240068`. Find at least two different values of `x` that can give this result.

(c) Compute the output of the following code snippet without using the computer. (You would need to use *Two's complement* to get hexadecimal representations of negative numbers – `https://bit.ly/3EZZ8US`.)

```
int a = 17;
int b = -41;
// bitwise AND
cout << hex << (a & b) << endl;
// bitwise OR
cout << hex << (a | b) << endl;
// bitwise XOR
cout << hex << (a ^ b) << endl;
// bitwise NOT
cout << hex << (~b) << endl;
```

1.(d). Analyze side-effects in operations, including arrays, increments, short-circuit Boolean evaluation.

(a)
```
int a[] = {1,3,5,7,9};
int i = 0;
if ((a[i++] % 2 == 0 && a[i++] - 3) || a[i++] % 3 == 2) { ... }
```
What is the end-state of the array after the expression in `if` statement is evaluated?

1.(e). Run pseudocode pointer operations on struct datatypes, draw arrows.

(a) We declare the doubly-linked node structure:

```
struct Node { int info; Node* prev; Node* next; }
```

Consider the following structure built from these nodes: Run the following pseu-

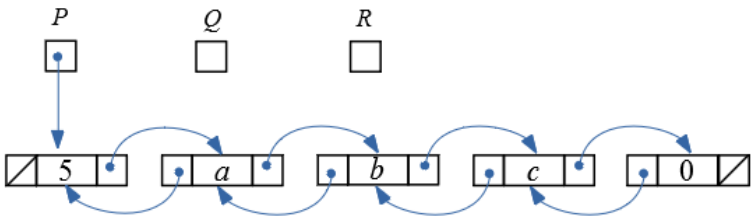

Figure 3: Pointers and Nodes.

docode and draw the nodes and pointer states after Line 7 and Line 12 of the code.

```
1  Q = P
2  Q = Q → next
3  R = Q → next
4  if Q → info ≤ P → info
5     R → prev = P
6  else
7     P → prev = R
   (Picture 1: Show the pointers at this point)
8  P → next → next = R → next
9  if R → info ≤ Q → info
10    R → next = P → next
11 else
12    R → next = P → prev
   (Picture 2: Show the pointers at this point)
```

1.(f). **(C++ code)** Perform bit manipulation.

(a) Assume that a cryptographer wants to transform every `char` into another `char` by applying bit rotation: three bits to the right. For example, the character `'C'` with its ASCII code `0x43` (as hex) or `01000011` (as binary) should be transformed into `01101000`, which is hex `68` or character `'h'`. Complete the function to transform:

```
char transform(char input) {
  // inswert function body to compute bit rotation
  char result = ...
  return result;
}
```

2 C++ with Object Orientation

2.(a) Passing parameters by value/reference, `const` modifier, default parameter values.

(a) The following code shows overloaded method `Square::square` with two different parameters – either one integer or one double. What happens, if you call this method on argument `'7'`?

```
#include <iostream>
using namespace std;
class Square {
  public:
  int square(int a) {  return (a*a); }
  double square(double b) { return b*b; }
};

int main() {
  Square ss;
  cout << ss.square('7') << endl;
}
```

(b) What are the values of local variables `a,b` at the end of functions `fun()` and `main()` respectively?

```
void fun(int a, int& b) {
  a += 10;
  b += 10;
  // What are the values of a, b here?
}
int main() {
  int a = 13;
  int b = 15;
  fun(++b,a);
  // What are the values of a, b here?
}
```