Midterm 1 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**
   - 1.A. Restore/drop parentheses, use syntax trees.
   - 1.B. Translate between flowcharts and C++ control structures.
   - 1.C. Use bit arithmetic.
   - 1.D. Side-effects in operators, arrays, short-circuit Boolean evaluation.
   - 1.E. Run pseudocode pointer operations, draw arrows.
   - 1.F. (C++ code) Perform bit manipulation.
   - 1.G. (C++ code) Manipulate arrays, char arrays (C-strings), 2D arrays.
   - 1.H. (C++ code) Input data using ”iostream”, ”sstream”, ”getLine”, ”get”, ”peek”
2. **C++ with Object Orientation**
   - 2.A. Parameter passing to functions, ”const” modifier, default parameters.
   - 2.B. The order how constructors and destructors are executed.
   - 2.C. (C++ code) Implement inheritance with virtual/non-virtual functions.
   - 2.D. (C++ code) Custom comparison function to generic sorting, max or similar algorithm.
3. **Big-O, Omega, Theta Notation**
   - 3.A. Find the asymptotic growth for a given function.
   - 3.B. Compare classes of function growth or order them.
   - 3.C. Express time complexity for recursively defined functions.
   - 3.D. Express time complexity for a code snippet “from the inside out”.
   - 3.E. Find the amortized time complexity for an operation on a given data structure.
   - 3.F. Count the number of calls for comparisons or similar functions.
4. **Lists, stacks, queues.**
   - 4.A. Use Abstract Data Type (ADT) to write algorithms.
   - 4.B. (C++ code) Use STL classes for lists, stacks, queues with iterators.

1 C++ apart from Object Orientation

1.A. **Restore/drop parentheses, use syntax trees.** Use this cheatsheet of operator precedence and associativity, if necessary: `https://bit.ly/3Bcigg4`.

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

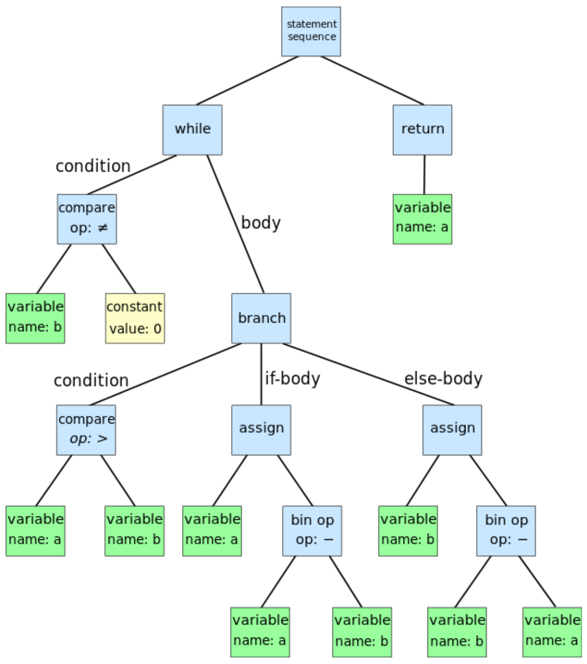

Figure 1: Syntax Tree

(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:

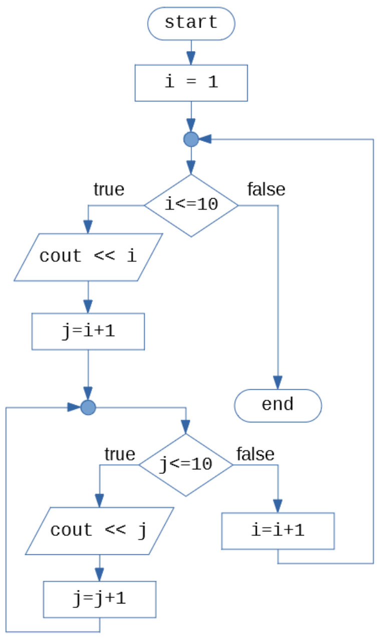

Figure 2: Syntax Tree

(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.** (Bitwise AND, OR, NOT, XOR, left/right shifts.)

(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. **Side-effects in operators, arrays, 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, 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-

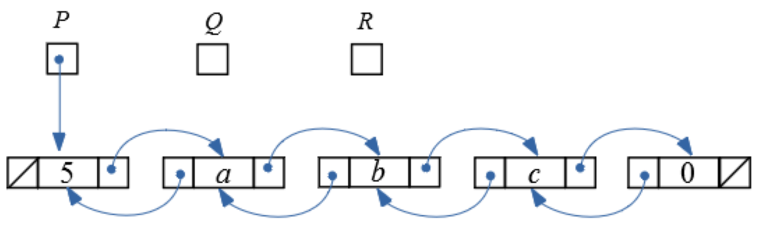

Figure 3: Pointers and Nodes.

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

```
1   $Q = P$
2   $Q = Q \rightarrow next$
3   $R = Q \rightarrow next$
4   **if** $Q \rightarrow info \leq P \rightarrow info$
5       $R \rightarrow prev = P$
6   **else**
7       $P \rightarrow prev = R$
    *(Picture 1: Show the pointers at this point)*
8   $P \rightarrow next \rightarrow next = R \rightarrow next$
9   **if** $R \rightarrow info \leq Q \rightarrow info$
10      $R \rightarrow next = P \rightarrow next$
11  **else**
12      $R \rightarrow next = P \rightarrow prev$
    *(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;
}
```

1.G. **(C++ code) Manipulate arrays, char arrays (C-strings), 2D arrays.**

(a) Using `char` pointers only implement function:
`int myStrcmp(const char * str1, const char * str2 );` It should read both strings simultaneously until it reaches terminating character `\0` in one of them.
It returns 0, if both strings are identical up to that character.
It returns -1, if string ”str1” is alphabetically before ”str2”. (or ”str1” is identical with ”str2”, but terminates sooner).
It returns 1, if string ”str1” is alphabetically after ”str2”. (or ”str1” is identical with ”str2”, but terminates sooner).

Do not use any predefined string functions, just the ”char” pointers and char comparisons. Test this function by the following code:

```
// Place the includes, namespaces and your function "myStrcmp" here

int main() {
  cout << myStrcmp("ABC", "ABC"); // prints 0
  cout << myStrcmp("ABC", "ABD"); // prints -1
  cout << myStrcmp("ABC", "ABCD"); // prints -1
  cout << myStrcmp("ABC", "ABB"); // prints 1
  cout << myStrcmp("ABC", "AB"); // prints 1
}
```

1.H. **(C++ code) Input data using "iostream", "sstream", "getLine", "get", "peek".**

Write a C++ program that reads standard input from STDIN (it stops reading when it encounters and empty line – to linebreaks in a row without any text). For each line the program inputs all the integers and adds them.

Sample input:

```
1 3 5
1 3 5 7 9 11

4 6
```

Sample output:

```
9
36
```

2 C++ with Object Orientation

2.A. **Parameter passing to functions, "const" modifier, default parameters.**

(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?
}
```

2.B. **The order how constructors and destructors are executed.**

(a) Consider the following code. List the order how the constructors and destructors of `Vaccination`, `Employee` and `Manager` are invoked. What is the value of `energyPoints` of the manager `m2` at the end of this program?

```
#include <string>

using namespace std;
class Vaccination {
    public:
    string type {};
    int year {};
};

class Employee {
    public:
    Vaccination v;
    int energyPoints;
    virtual void energize() {
        energyPoints += 100;
    }
};

class Manager: public Employee {
    public:
    Manager(Vaccination v, int energyPoints) {
        this -> v = v;
        this -> energyPoints = energyPoints;
    }
    void energize() {
        energyPoints += 123;
    };
};

void energizeTwice(Employee* e) {
    e -> energize();
    e -> energize();
}

int main() {
    Manager m1 { {"Comirnaty", 2021}, 150};
    Manager m2 = m1;
    m1.energize();
    energizeTwice(&m2);
    return 0;
}
```

2.C. **(C++ code) Implement inheritance with virtual/non-virtual functions.**

(a) What is the `width` and `height` of both the squares `s1` and `s2` at the end of the program:

```
#include <iostream>
#include <string>

using namespace std;
class Rectangle {
    protected:
    int width, height;
    public:
    string color;
    Rectangle(int width, int height) {
        this->width = width;
        this->height = height;
        color = "gray";
    }

    void makeWider() {
        width *= 2;
    }
};

class Square: public Rectangle {
    public:
    Square(int size): Rectangle(size, size) {
        color = "red";
    }
    void makeWider() {
        width *= 2;
        height *= 2;
    }
};

int main() {
    Square s1(17);
    Square s2 = s1;
    s1.makeWider();
    Rectangle* r2 = &s2;
    r2 -> makeWider();
}
```

2.D. **(C++ code) Custom comparison function to sorting, max or similar algorithm.**

Finish the implementation of method main: It should read in student ages and heights and output the "maximal student" – the student who has the largest age (if there are multiple students of the same age, output the tallest one).

```
#include <iostream>
#include <iomanip>
#include <set>

using namespace std;
```

```
struct Student {
  int age;
  double height;
  Student(int aa = 1, double hh = 1):
          age(aa), height(hh) {}

  friend istream &operator>>(
    istream  &input, Student &S ) {
      input >> S.age >> S.height;
      return input;
  }

  friend ostream &operator<<(ostream &output, const Student &S ) {
      output << "Student(" << S.age << "," << std::fixed <<
          std::setprecision(5) << S.height << ")";
      return output;
  }

  friend bool operator<(const Student &left, const Student &right) {
      return (left.age<right.age) ||
  (left.age == right.height && left.height<right.height);
  }
};

int main() {
  // read in 10 students into a STL class "vector"
  // find the maximum student (the tallest among the eldest ones)
  // output that maximum student to "cout".
}
```

3 Big-O, Omega, Theta Notation

3.A. **Find the asymptotic growth for a given function.**

(a) Define the following function $f(n)$

$$f(n) = n^{\log_{10} n} + (1 + \sin(n))n^{10}.$$

Find optimal asymptotic $g_1(n)$ (upper estimate) and $g_2(n)$ (lower extimate). Namely, we should have $f(n)$ is in $O(g_1(n))$ and $f(n)$ is in $\Omega(g_2(n))$.

3.B. **Compare classes of function growth or order them.**

(a) Rank the following functions so that for every two functions $g_i(n)$ and $g_j(n)$ in this ordering, $g_i(n)$ is to the left of $g_j(n)$ iff $g_i(n)$ is in $O(g_j(n))$.

$$g_1(n) = n^e, \quad g_2(n) = e^n, \quad g_3(n) = \binom{n}{5}, \quad g_4(n) = \sqrt{2^{\sqrt{n}}},$$

$$g_5(n) = \binom{n}{n-4}, \quad g_6(n) = 2^{5(\log_2 n)^3}, \quad g_7(n) = n^{(\log_2 n)^2}, \quad g_8(n) = n^3 \cdot \binom{n}{3}.$$

3.C. **Express time complexity for recursively defined functions.**

(a) Find some $\Theta(g(n))$ containing the function satisfying the following recurrence:

$$\begin{cases} T(n) = 1, \text{if } n \leq 2 \\ T(n) = T\left(\left\lceil \frac{n}{3} \right\rceil\right) + T\left(\left\lceil \frac{2n}{3} \right\rceil\right) + n. \end{cases}$$

This would be the complexity of an algorithm that receives input of length n, splits into two unequal parts (of sizes $\lceil n/3 \rceil$ and $\lceil 2n/3 \rceil$) and then applies itself recursively (and finally spends n time to combine both computations together).

(b) Find an asymptotic solution of the following recurrence:

$$\begin{cases} T(n) = 1, \quad \text{if } n = 0 \text{ or } n = 1, \\ T(n) = \log_2 n + T(\sqrt{n}), \quad \text{if } n > 1. \end{cases}$$

Express your answer using Θ-notation, and justify your answer.

3.D. **Express time complexity for a code snippet "from the inside out".**

(a) `a[i][j]` is an $n \times n$ matrix with integer numbers, and here is a function to transform that matrix. Find the time complexity of this function as $\Theta(g(n))$.

```
void transform (int** a, int n) {
  for (int i = 0; i < n - 1; i ++) {
    for (int j = i+1; j < n; j++) {
      int tmp = a[i][j];
      a[i][j] = a[j][i];
      a[j][i] = tmp;
    }
  }
}
```

3.E. **Find the amortized time complexity for an operation on a given data structure.**

(a) Somebody is incrementing a 10-digit counter from its initial value `0000000000` to the final value `9999999999`. Changing one digit to a new value in that counter takes 1 unit of energy. In general, changing k digits takes k units of energy.

For example, incrementing from `0000000013` to `0000000014` takes 1 unit of energy (only the last digit changes). On the other hand, incrementing from `0999999999` to `1000000000` takes 10 units of energy (as all the digits change).

Find the amortized cost per one increment as the counter moves from the state `0000000000` to the final state `9999999999`.

3.F. **Count the number of calls for comparisons or similar functions.**

(a) Consider the following code to find the 2nd smallest element in an array of objects `arr[n]` of class `X`, where $n > 2$.

```
bool lessThan(X a, X b) {
    // some comparison function
}

void swap(X& a, X& b) {
    X temp = a;
    a = b;
    b = temp;
}

X getSecondLargest(X* arr, int n) {
    X smallOne = arr[0];
    X smallTwo = arr[1];
    if (lessThan(smallTwo, smallOne))
        swap(smallOne,smallTwo);
    for (int i = 2; i < n; i++) {
        if (lessThan(arr[i], smallTwo))
            smallTwo = arr[i];
        if (lessThan(smallTwo, smallOne))
            swap(smallOne,smallTwo);
    }
    return smallTwo;
}
```

What is the worst-case number of comparisons (calls to `lessThan(a,b)` when finding the second smallest element in an array of length n. (Try to find the exact count of comparisons; or at least estimate their number with Big-O notation.) Would it be better to use an efficient sorting algorithm (such as `MergeSort`) on the array, and then pick the second smallest element in the sorted list?

4 Lists, stacks, queues

4.A. **Use Abstract Data Type (ADT) to write algorithms.**

(a) Given two sorted lists, L_1 and L_2, write a pseudocode to compute $L_1 \cap L_2$ using only the basic list ADT operations. The arguments L_1 and L_2 can be modified or erased in the process.

(b) Given list L_1 return the size of it.

(c) Given list L_1 and value x, test if value x is contained in the linked list

(d) Given list L_1 and a value x, add x to L_1, if it is not already contained in the linked list.

(e) Given list L_1 and a value x, remove x, if it is contained in the linked list.

4.B. **(C++ code) Use STL classes for lists, stacks, queues with iterators.**

(a) There are two instances of STL list (containing integers). Both of them are represented in non-decreasing order. Write a function to merge them into a single STL list (so that it contains the same elements as in both lists – in common sorted order).

```
// your implementation of "merge(aa,bb)"

int main() {
  list<int> aa = { 1,4,7,11  };
  list<int> bb = {2,3,5, 8, 13 };
  list<int> cc =  merge(aa,bb);
  // cc should be { 1, 2, 3, 4, 5, 7, 8, 11, 13 }
}
```