Exam 1

Data Structures
Thursday, October 7, 2021

You must justify all your answers to recieve full credit

---

1. Consider a code fragment using the bitwise XOR.

```
int a = -3;
int x = ???  // replace this ??? with a number
int b = a^x;
cout << hex << b;
// This outputs the hexadecimal representation of "b": "ffffabcd".
```

   (a) Write the hexadecimal representation of variable `a`, if its decimal value is -3.

   (b) Write the binary representation of the same variable `a`.

   (c) Write the hexadecimal representation of variable `x`. That is, rewrite line 2 of the code snippet above to make variable `b` equal to `0xffffabcd`.

2. A doubly linked list has 3 structures of type `Node` (see Figure 1): it has `prev` and `next` pointers of type `Node*` and also a constant `info` field storing constant positive integers. That is, the `info` field does not change over the lifetime of the `Node`.

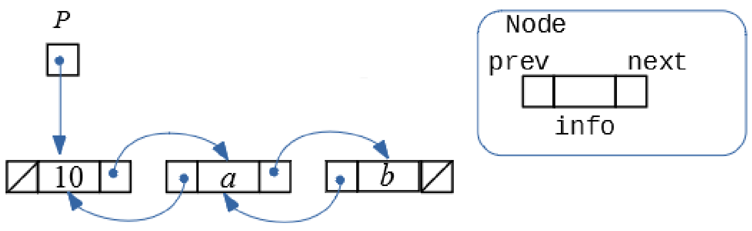

Figure 1: Doubly linked list

Variable `P` is a pointer of type `Node*`, initially `P` points to the first node in this list. The `info` fields in this list have values 10, a and b.

Write code that modifies this doubly linked list in the following way:

- If $a > b$, the pointers in the list change so that the `Node` with `info` field value 10 points to b (that `Node` is now second), and the `Node` with `info` field b points to a (that `Node` is now third).
- If $a \leq b$, then your code should leave the list unchanged.

3. The following C++ program declares a class `Pair`. Pairs are *lexicographically ordered*:

$$(x_1, y_1) < (x_2, y_2) \text{ iff } x_1 < x_2 \lor (x_1 = x_2 \land y_1 < y_2).$$

Complete the code below so that, when compiled and executed, in order:

- a positive integer n is input,
- n pairs from the standard input are input,
- the n pairs are pushed on the stack, skipping pairs which are not lexicographically larger than the current top element of the stack,
- the remaining pairs are output to the standard output as separate lines in their original order.

Use the overloaded operators "`cout << pair`", "`cin >> pair`", "`p1 < p2`" for input, output and comparisons. The only data structure to use is STL stack. If necessary, you may use several stacks.

```
#include <iostream>
#include <stack>
using namespace std;
class Pair { public: int x; int y; };
istream &operator>>(istream  &input, Pair &p ) {
  input >> p.x >> p.y;   return input;
}
ostream &operator<<(ostream &output, const Pair &p ) {
  output << "(" << p.x << "," << p.y << ")";   return output;
}
bool operator<(const Pair &left, const Pair &right) {
  // implement the lexicographic comparison operator.
}
int main() {
  // Input the total number of pairs, then 2*n integers (the pairs).
  // Output those pairs which are in lexicographically increasing order.
}
```

Sample input

```
5
4 17
4 17
5 1000
7 12
7 9
```

Sample output

```
(4,17)
(5,1000)
(7,12)
```

4. In your code from Question 3, define the default (no argument) constructor, copy constructor and destructor. Describe the order in which these are called when your compiled code is executed with the input below.

```
2
2 3
1 4
```

In your answer you should list the order how the constructors of both types (and also destructor) are invoked for every `Pair` object. Describe why this behavior makes sense and generalize – how many invokations would happen for n pairs in the input.

5. Consider the functions $f_1(n)$, $f_2(n)$, $f_3(n)$, $f_4(n)$ below, mapping positive integers $n \geq 5$ to positive real numbers $t > 0$:

$$\begin{aligned} f_1(n) &= (1 + \cos n)\sqrt{2^{7\cdot\log_2(n)}}, \\ f_2(n) &= 13^{\log_2(n)}, \\ f_3(n) &= \frac{1}{n^2} \cdot \binom{n}{5}, \\ f_4(n) &= f_1(n) + f_2(n) + f_3(n). \end{aligned}$$

For each function $f_i(n)$, $i = 1, 2, 3, 4$, find functions

- $g_i(n)$ such that $f_i(n)$ is $O(g_i(n))$,
- $h_i(n)$ such that $f_i(n)$ is $\Omega(h_i(n))$,
- $k_i(n)$ such that $f_i(n)$ is $\Theta(k_i(n))$.

Give justifications for all your answers.