

CMPT 225-D100 Midterm Exam 1

Summer 2023, Burnaby

Sample Exam

This is a **50 minute closed book exam**: notes, books, computers, calculators, electronic devices, etc. are **not** permitted. Do not speak to any other students during their exam or look at their work. Please remain seated and **raise your hand** if you have a question.

Linked Lists

(10 marks) Suppose you have a C++ program with a **singly-linked list** based on this:

```
struct Node {
    int data;
    Node* next;
};

Node* head = nullptr; // head is a global variable
```

head is a global variable that points to the first element of the list. If head is nullptr, then the list is empty. Note that there is *no* class, and the `negate_biggest()` function requested below should directly use head.

Question

Using **detailed C++-like pseudocode**, write a function called `negate_biggest()` that:

- Returns a copy of the biggest data value in the list.
- Negates (i.e. multiplies by -1) the biggest value in the list. No other changes are made to the list.
- Traverses the list at most one time.
- **Doesn't** use recursion.

Assume the list is non-empty, and that there are *no* duplicates (i.e. every data value is unique).

For example, suppose your program creates a list like this:

```
head = new Node{0, nullptr};
head = new Node{2, head};
head = new Node{1, head};
```



When you call `negate_biggest()` it returns 2, and the 2 in the list is negated:



Your `negate_biggest()` should work for *any* valid singly-linked list, not just this example. Your answer should be efficient and not use any unnecessary memory. **Don't** use arrays or vectors or other such data structures in your answer.

Please write your answer on the next page.

Sample Exam

Stacks

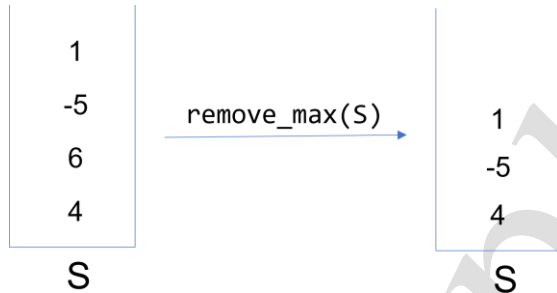
(10 marks) Suppose the `Stack` class contains `ints`, and has these methods:

- `S.push(x)` inserts `x` on the top of `S`.
- `S.pop()` removes, but does **not** return, the top `int` of `S`; if `S` is empty, an error occurs.
- `S.peek()` returns, but does **not** remove, a copy of the top `int` of `S`; if `S` is empty, an error occurs.
- `S.empty()` returns `true` if `S` is empty, and `false` otherwise.
- `S.size()` returns the number of `ints` in `S`.
- When you define a new stack it starts empty, e.g.:
`Stack S; // S is empty`
`Stack T; // T is empty`

Question

Using **detailed C++-like pseudocode**, write a function (not a method!) called `remove_max(Stack& S)` that removes the *biggest* element in `S`, and leaves the rest of the elements in the same order. To make things easier, you can assume `S` is non-empty, and that all the `ints` on `S` are different (i.e. `S` has no duplicates).

For example:



Important Only use `Stacks` in your answers: **don't** use arrays, vectors, lists, or other such data structures. You can use as many stacks as you need (but try to use as few as possible).

Important `remove_max(Stack& s)` is a *function*, **not** a method in a class. So it can only use the `Stack` methods described above, and should not make any assumptions about how `Stack` is implemented.

Write your answer on the next page.

Please use brief comments to explain the main sections of your answer.

Sample Exam

```
// Pre-conditions:  
//   S is not empty  
//   S has no duplicates  
// Post-condition:  
//   modifies S so that its max element is removed,  
//   and other elements are still there in the same order  
  
void remove_max(Stack& S)  
{
```

Sample Exam

O-notation and Analysis

a) (5 mark) State the precise mathematical definition of “ $f(n)$ is $O(g(n))$ ”.

b) (5 marks) Using the definition of O-notation, mathematically prove that $20n - 50$ is $O(n)$.

Trees

Let p be a node in a tree T , as defined in the textbook and lectures. The **depth** of p is defined to be the number of ancestors of p , *excluding* p itself.

a) (4 marks) Draw a tree:

- that is **not** binary.
- where the root is labelled “root”.
- with **exactly** 4 **leaf** nodes, each labelled “leaf”.
- with **exactly** 2 different nodes of **depth** 4, labelled A and B.

b) (2 marks) Give a **recursive definition** of the depth of a node p in a tree T .