The University of Michigan
Electrical Engineering & Computer Science
EECS 281: Data Structures and Algorithms
Fall 2024

Lab 2: Arrays, Linked Lists, Stacks, Queues, Deques for 281

Instructions:
Work on this document with your group, then enter the answers on the Canvas quiz.

Note:
Be prepared before you meet with your lab group, and read this document so that you know what you must submit for full credit. You can even start it ahead of time and then ask questions during any lab section for help completing it.

You MUST include the following assignment identifier at the top of every file you submit to the autograder as a comment. This includes all source files, header files, and your Makefile (if there is one). If there is not autograder assignment, you may ignore this.

Project Identifier: 1CAEF3A0FEDD0DEC26BA9808C69D4D22A9962768

---

1 Logistics & Slide Comprehension (1 point)

1. When is lab 2's quiz due?
   - A. 9/23/2024
   - B. 9/22/2024
   - C. 9/21/2024
   - D. 9/24/2024
2. When is the lab 1 autograder and quiz due?
   - A. 9/25/2024
   - B. 9/24/2024
   - C. 9/23/2024
   - D. 9/16/2024
3. When is the lab 2 handwritten problem due?
   - A. 9/24/2024
   - B. 9/16/2024
   - C. 9/10/2024
   - D. 9/18/2024
4. Which of the following was definitely NOT covered in this lab?
   - A. Arrays
   - B. Queues and Stacks
   - C. Complexity Analysis
   - D. Hash Tables

2 Complexity Analysis (2 points)

5. Consider the complexity of the following function:

$$f(n) = 3n^6 - 3n^2 + 4n + 2$$

Which of the following statements are true?

A. $f(n) = O(3n^2)$
B. $f(n) = O(3n^6 - 3n^2)$
C. $f(n) = O(2n^3)$
D. $f(n) = O(n^7)$
E. $f(n) = O(n^n)$

6. What is the complexity of the following algorithm? x, k, y, and s correspond to the sizes of vecX, vecK, vecY, and vecS respectively. Pick the most accurate and precise answer.

```
for (int x : vecX) {
    for(int k: vecK){
        cout << "constant time work\n";
    }
}
for (int y : vecY) {
    for(int s: vecS){
        cout << "more constant time work\n";
    }
}
```

A. $\Theta(x + s)$
B. $\Theta(xk + ys)$
C. $\Theta(x)$
D. $\Theta(y)$
E. None of the above!

7. What is the runtime complexity of the following function?

```
int lets_count(int m, int n) {
    int count = 0;
    while (n > 1) {
        for (int i = 1; i < m; i*= 5)
            count++;
        n /= 3;
    }
}
```

A. $\Theta(m \log n)$
B. $\Theta(n \log m)$
C. $\Theta(\log(m) \log(n))$
D. $\Theta(\log mn)$

8. What is the runtime complexity of the following code snippet?

```
for (int i = 1; i < n; ++i) {
    for (int j = 1; j < log(n); j *= 2) {
        cout << '-';
    }
}
```

A. $\Theta(n \log n)$

B. $\Theta(n \log(\log(n)))$

C. $\Theta(\log^2(n))$

D. $\Theta(n^2)$

3 Arrays, Linked Lists, Stacks, and Queues (2 points)

9. Which of the following are true regarding Linked Lists and Arrays?
   - A. Arrays are allocated in a single contiguous chunk in memory
   - B. Linked lists are allocated in non-contiguous chunks in memory
   - C. The distance (in number of elements) between two elements in an array can be found in $\Theta(1)$ time for arrays, but only $\Theta(n)$ for lists
   - D. All of the above are true

10. Which of the following regarding Arrays are true?
    - A. Random Access can be done in O(1) time
    - B. Inserting a new element into an array can take O(n) time
    - C. If an array is too small, you will have to reallocate memory
    - D. All of the above are true

11. Which of the following regarding Linked Lists are false?
    - A. Random Access can be done in O(n) time
    - B. Appending to the end of a linked list (with no tail pointer) will takes O(n) time
    - C. One way that you can keep track of the number of elements in a linked list is having a size variable
    - D. Linked lists are allocated in a single chunk of contiguous memory

12. Which of the following regarding Stacks, Queues, and Deques are false?
    - A. The worst case time complexity of sorting a stack using only an auxiliary stack and O(1) additional space is $O(n^2)$ (see slide 46).
    - B. With the approach for implementing a Queue with Stacks discussed in the lab slides (slide 75), the worst case time complexity for removing an element is O(n)
    - C. A Deque can be used to replicate the functionalities of both stacks and queues
    - D. All of the above are actually true!

13. Which data structure is preferable for storing large data types that often must be inserted at random positions? For the linked list option, assume you have a pointer to the element where you want to insert.
    - A. array
    - B. linked list
    - C. both arrays and linked lists work equally well

14. What is the time complexity of searching for an element in an unsorted array? In an unsorted singly-linked list? In a sorted array? In a sorted singly-linked list? Consider the most efficient algorithm for each search.
    - A. unsorted array: $\Theta(log\ n)$, unsorted linked list: $\Theta(n)$, sorted array: $\Theta(log\ n)$, sorted linked list: $\Theta(log\ n)$
    - B. unsorted array: $\Theta(log\ n)$, unsorted linked list: $\Theta(n)$, sorted array: $\Theta(log\ n)$, sorted linked list: $\Theta(n)$

C. unsorted array: $\Theta(n)$, unsorted linked list: $\Theta(n)$, sorted array: $\Theta(log\ n)$, sorted linked list: $\Theta(log\ n)$

D. unsorted array: $\Theta(n)$, unsorted linked list: $\Theta(n)$, sorted array: $\Theta(log\ n)$, sorted linked list: $\Theta(n)$

15. What runtime is needed to remove an element at the bottom of a stack with n elements and return the stack with all of the other elements in their original order?

    A. $\Theta(1)$
    B. $\Theta(n)$
    C. $\Theta(n^2)$
    D. $\Theta(n^3)$

16. What is the additional space complexity of removing an element at the back of a queue with n elements and returning the queue with all other elements in their original order?

    A. $\Theta(1)$
    B. $\Theta(n)$
    C. $\Theta(n^2)$
    D. $\Theta(n^3)$

4 Abstract Data Types (1 point)

17. Suppose you are an adventure seeker interested in visiting the most exciting destinations around the world. Whenever you hear of a new place you want to visit, you add it to the bottom of a list you maintain. When you want to travel, you scratch off the top item from your list and travel there. Which of the following abstract data types can be efficiently used for this task? There may be multiple correct answers!

    A. deque

    B. vector

    C. stack

    D. queue

    E. none of the above

18. Suppose you are an exam grader in charge of grading an exam. You are grading while the exam is in progress. Whenever you are done grading an exam, you take the next exam off a pile of exams on your table. However, since students will be taking the exam while you are grading, they will occasionally come to you to turn in their exams. Each student will place their completed exam on top of the pile of exams on your table. Which of the following abstract data types can be efficiently used to simulate the order in which the exams are graded? There may be multiple correct answers!

    A. deque

    B. vector

    C. stack

    D. queue

    E. none of the above

5 Optimization Tips (1 point)

19. Your code is running over time on the autograder. Which of the following could you do to speed up your code (the performance improvement should be non-negligible)? Select all that apply. There may be multiple correct answers!

    A. you could replace return 0 with exit(0)in main(), since exit(0)does not call container destructors

    B. you could pass all primitive types by reference instead of by value

    C. because function calls are very expensive in C++, you could inline all your code by hand (that is, instead of calling functions, you would simply copy and paste in the code you need)

    D. if you have any objects without member functions, you could declare them as structs instead of classes

    E. you could replace endl with \n, especially if your program prints out a lot of output

20. You have an input file that you are trying to read in. The first line of this file contains a number n, and the following n lines contain words (one word per line). You read in the number n using operator>>, reserve a vector of strings to capacity n, and push the following n words in the input file into this vector (calling getline until all lines of the input file have been read). After your program runs to completion, you realize that your vector's capacity is twice as large as it should be. Which of the following could be a reason why? There may be multiple correct answers!

    A. you are using vector .push_back() instead of operator[] when adding the list of words to your vector

    B. you are using operator[] instead of vector .push_back() when adding the list of words to your vector

    C. you may have pushed back a space or newline character with the first getline call after using operator>>

    D. the input file may have only contained n - 1 words instead of n words

    E. none of the above

6 Handwritten Problem

This problem is to be submitted independently. We recommend trying it on your own, checking your answer with your group and discussing solutions, and then writing down a final form to submit. These will be graded on completion, not by correctness. However, we want to see that you were thinking about the problem.

21. Linked Queue: Implement the interface shown below for a queue that uses a singly linked list to represent its data. The list had a head pointer, which points to the head of the linked list, and a tail pointer that points to the tail-end of the linked list. It also has a count variable which keeps track of the number of nodes in the list.

```
template <typename T>
struct Node {
    T     value;
    Node* next;
};


template <typename T>
class LinkedQueue {
private:
    Node<T>* head  = nullptr;
    Node<T>* tail  = nullptr;
    size_t   count = 0;
public:
    T front() const { /* Implement */ }
    void pop() { /* Implement */ }
    void push(T x) { /* Implement */ }
    size_t size() const  { return count; }
    bool empty() const { return count == 0; }
    ~LinkedQueue() { /* Implement */ }
};
```

7 Coding Assignment

You can work on this problem by yourself or with your group, but a solution must be submitted to the autograder for each individual.

The *fixity* of an operator describes its position relative to its operands. A *prefix* operator precedes its operands and a *postfix* operator follows them. In the case of operators that operate on exactly two operands, we can also say that an *infix* operator appears between its operands. An example of each:

-3	The one-operand minus sign for negation is a prefix operator
$4!$	The exclamation mark for factorial is a postfix operator
$5+6$	The plus sign for addition is an infix operator

Your task, however, is to evaluate expressions written in *anyfix* notation, meaning any operator can act as a prefix, postfix, or infix operator depending on context. For consistency across operators, you need only handle these four:

$+$	Addition
$-$	Subtraction
$*$	Multiplication
$/$	Integer division

All four take exactly two operands. There is no negative sign; $-$ always means subtraction.

What does "depending on context" mean? As an expression is evaluated from left to right:

- When an operator is encountered that does not immediately have two operands waiting for it, it will wait until the second operand arrives before it operates. Examples:
  - In $+$ 1 2, the addition operator waits for 1 and 2 and then adds them
  - In 3 $*$ 4, the multiplication operator has 3 waiting but must wait for 4 before it can perform multiplication
- When a number is encountered that does not immediately cause a waiting operator to have two operands, it will wait. Examples:
  - In 3 $*$ 4, the 3 waits and is later operated on by the multiplication operator
  - In 5 6 $-$, both 5 and 6 wait and are later operated on by the subtraction operator

When an operator operates on operands, it produces a result that is immediately encountered. This means an expression like 1 $-$ 2 $*$ 3 will be evaluated how we would ordinarily evaluate $(1-2)*3$.

More-recently-encountered operators and operands take precedence. This means the expression 3 2 1 $-$ / is evaluated by subtracting 1 from 2, and then dividing 3 by the result. Similarly, $*$ $+$ 4 5 6 is evaluated by adding 4 to 5, then multiplying the result by 6.

When evaluation of an anyfix expression finishes, there will be exactly zero operators and one number waiting. The result of evaluation is that number.

You must write a function that accepts a single well-formed anyfix expression and returns the result of evaluating that expression. A well-formed anyfix expression consists of n digits and $n-1$ operators in any order, and nothing else. Not even whitespace.

Each digit represents a separate number. 12 means one followed by two, not twelve.

Your implementation must be in the form of a function matching this prototype:

```
std::int64_t evaluate(std::string const& expression);
```

Place this function in a file named `evaluate.cpp`. Submit your implementation on the EECS 281 autograder, to the assignment titled "S24 Lab 2 - Anyfix Notation".

Some hints:

- Use one or more of the data structures discussed in recent lectures and labs.
- Write plenty of tests of your own. A file is provided in the starter files for this.
  - Be conscious of your own procedure for solving the tests you write!
- Run `make help` for a helpful overview of the capabilities of the makefile we provide.

Make sure that all files submitted to the autograder include the project identifier listed on the first page of this assignment.

Project identifier: 1CAEF3A0FEDD0DEC26BA9808C69D4D22A9962768