CSCI-1200 Data Structures
Test 1 — Practice Problems

Note: This packet contains selected practice problems from Test 1 from three previous years. Your test will contain approximately one third to one half as many problems (totalling ∼100 pts). Problems may be extensions to specifc homework assignments used that year.

1 Carpet Warehouse Orders [35 pts]

Congratulations! You've just been offered a Summer Arch internship at 1-800-CARPETS! Here's a sample of customer orders for wall-to-wall carpet received on a typical day:

```
std::vector<Carpet> orders;
orders.push_back(Carpet(12, "white",   3.5));
orders.push_back(Carpet(15, "white",  19.5));
orders.push_back(Carpet(15, "blue",   10.0));
orders.push_back(Carpet(12, "silver", 10.5));
orders.push_back(Carpet(12, "white",   8.5));
orders.push_back(Carpet(15, "blue",   20.0));
```

The carpet is manufactured in large rolls in one of two widths: 12' or 15' (measured in feet), and is available in a wide variety of colors. The necessary length of carpet will be cut from one of these large rolls to fit the dimensions of the customer's room. The customer is charged based on the area (square feet).

After collecting and organizing the orders for the day, the sales staff print information about the carpet rolls that will need to be in stock at the warehouse to satisfy the customer orders. Note that orders for the same color and width have been combined, and that the table is neatly formatted.

```
   blue, 15' roll,  450.0 sq feet
 silver, 12' roll,  126.0 sq feet
  white, 12' roll,  144.0 sq feet
  white, 15' roll,  292.5 sq feet
```

1.1 `Carpet` Class Declaration [11 pts]

First, write the class declaration or *header file* (`carpet.h`) for the `Carpet` object. Use `const` and reference where appropriate. To demonstrate the C++ syntax necessary for more complex classes, please save the implementation of ALL functions (even one-liners) for the *class implementation* on the next page.

sample solution: 12 line(s) of code

1.2 `Carpet` Class Implementation [12 pts]

Next, implement all `Carpet` member functions (and any related non-member functions) as they would appear in the `carpet.cpp` file.

sample solution: 16 line(s) of code

1.3 Printing the Necessary Warehouse Stock [12 pts]

Finally, complete the code fragment below to print the carpet data in the `orders` variable used in our initial example. Closely study the desired sample output on the first page; however, your code should work for any data, not just the specific example shown above!

```
// organize the data

std::sort(                                                                    );


// initialize a few helper variables









// loop over the orders and print each necessary carpet roll
for (unsigned int i = 0; i < orders.size(); i++) {

















}
```

sample solution: 16 line(s) of code

In your answer above, you are encouraged to write and use a helper function named `PrintCarpetRoll` to format each carpet roll in the table using the STL `iomanip` library. Implement that function below:

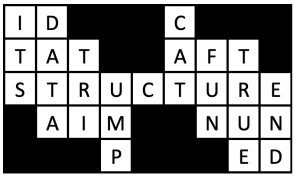
```

```

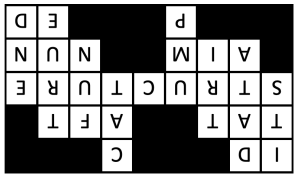
sample solution: 5 line(s) of code

2 Analyzing Crossword Symmetries [20 pts]

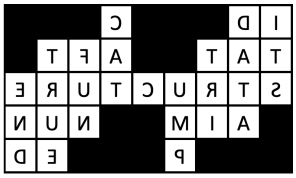
It is a common convention for the black squares of a crossword puzzle to be *rotationally symmetric* - when spun around 180°, the shapes formed by the black squares will be in the same positions. Alternatively, puzzles may be *horizontally-mirrored* – black squares are in the same positions if flipped left to right.



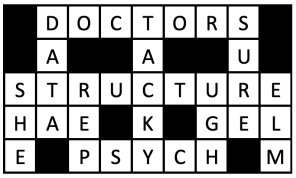
puzzle1



puzzle1 is *rotationally-symmetric*

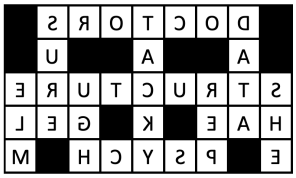


puzzle1 is *NOT horizontally-mirrored*



puzzle2

puzzle2 is *NOT rotationally-symmetric*



puzzle2 is *horizontally-mirrored*

Write a function named `check_symmetries` that takes three arguments, an STL `vector` of STL `strings` named `board`, and two boolean variables: `rotationally-symmetric` and `horizontally-mirrored`. The function should analyze the symmetries of the board and set the boolean variables appropriately.

sample solution: 19 line(s) of code

3 `ben++`, the Better `clang++`? [24 pts]

Ben Bitdiddle has decided to write a better C++ compiler named `ben++`. He's asked you to write the compilation configuration parser, which has similar options and format to both `g++` and `clang++`, except that the configuration is read from a file, not the command line. Furthermore, Ben's compiler can be parallelized: if `-p` is specified, the next value is an integer number of threads for parallel computation.

Here are a few example valid compilation configuration files:

```
ben++ main.cpp date.cpp name.cpp -Wall -Wextra -p 4 -o lab_executable.out
```

```
ben++ main.cpp
```

```
ben++ -Wall date.cpp -o lab_executable.out name.cpp -Wextra -p 2 main.cpp
```

Ben would like his compiler to catch some common programmer mistakes. For example, parsing:

```
ben++ -Wall -Wextra -p 4 -o a.out
```

should print to `std::cerr` the message: "`ERROR: No source code files provided!`". And

```
ben++ main.cpp date.cpp date.h -Wall -Wextra -p 4 -o lab_executable.out
```

should result in this `std::cerr` message: "`WARNING: You should not directly compile a .h file!`".

3.1 More Error Checking is More Better! [4 pts]

There are many other compiler configuration typos or misunderstandings that a novice C++ programmer can experience when using `g++` or `clang++`. Have you or one of your Data Structures lab friends made compilation configuration mistakes? Give a specific example of another buggy compilation configuration. In 1-2 well-written sentences describe the mistake with your example. What descriptive error or warning message could be printed to `std::cerr` that would help a programmer fix this specific mistake?

3.2 Compilation Parsing Implementation [20 pts]

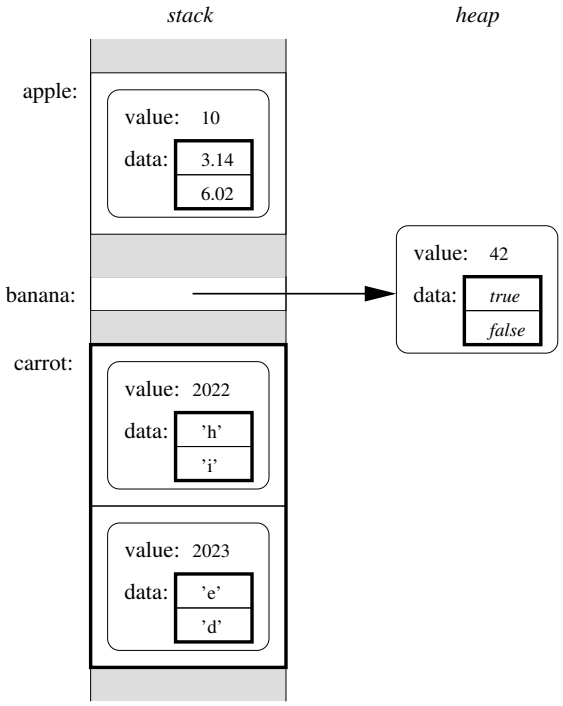
Write a fragment of code below to parse a compilation configuration file from the `istr` file stream variable. Once you've finished reading the file, call Ben's `DO_COMPILATION` function, which takes 4 arguments: an STL `vector` of STL `string`s that are the source code files, an STL `vector` of STL `string`s that are the compilation warnings to be enabled, the integer number of parallel threads (should default to 1), and an STL `string` indicating the executable filename (should default to `"a.out"`).

```
std::ifstream istr("compilation_configuration.txt");
```

sample solution: 27 line(s) of code

4 Object Memory [18]

The memory diagram below uses a class named `Object`. Write the class declaration and implement all member functions as they would appear in the `object.h` file.



sample solution: 15 line(s) of code

Finish the code fragment below using your `Object` class to create this memory diagram.

`apple;`

```
apple.setValue(10);
apple.setData(3.14, 6.02);
```

`banana;`

sample solution: 8 line(s) of code

5 Palindrome Art [15 pts]

Write a function named `palindrome` that will take in two arguments: the ASCII art message output from our Homework 1 as an STL `vector` of STL `string`s, and the integer width of a single letter (width=7 for this font). The function should return true if the message is a palindrome (a word containing the same letters when reversed), and false otherwise.

```
.............................................
.###....###....###....###....###....###....###...
#..........#..#......#...#..#..........#..#......
#.......####..#......####...#.......####..#......
#......#...#..#......#......#......#...#..#......
#.......####...###....####...###....####..#......
.............................................
```

```
.........................................
.###...####....####...###....####...###...
#...#..#...#..#...#......#..#...#..#...#..
####...#...#..#...#...####..#...#..####...
#......#...#...####..#...#...####..#......
.####..#...#......#...####......#...####..
...............###..........###..........
```

For example, the function should return true when given the top message. And false when given the bottom message. *Note: There is no kerning in this problem. (That was extra credit.)*

sample solution: 14 line(s) of code

6 Classic Trees [42 pts]

Over the pandemic, Louis B. Reasoner worked with RPI Environmental & Site Services to overhaul their system for tracking maintenance tasks on the arboreal specimens (a.k.a., trees) on campus. Here is an example of how the staff will interact with the tracking system. The following commands might appear in the `main` function. First, we can create a few specific trees and the year they were planted. The system allows us to schedule routine maintenance tasks for each tree.

```
Tree tallest("oak", 1950);
bool success = tallest.addMaintenance("fertilize");
assert (success == true);
Tree favorite("maple", 1995);
bool success = favorite.addMaintenance("water");
assert (success == true);
```

Next we use an STL `vector` container to store these and other trees:

```
std::vector<Tree> trees;
trees.push_back(favorite);
trees.push_back(tallest);
trees.push_back(Tree("pine",2002));
trees.push_back(Tree("oak",2012));
trees.push_back(Tree("ash",1990));
trees.push_back(Tree("maple",2015));
trees.push_back(Tree("maple",2008));
```

Below we schedule maintenance tasks to prune all trees that were planted in `2005` or earlier, to treat all `ash` trees for invasive insects, and to water all `maple` trees.

```
int prune_tasks = addMaintenance(trees,2005,"prune");
assert (prune_tasks == 4);
int insecticide_tasks = addMaintenance(trees,"ash","insecticide");
assert (insecticide_tasks == 1);
int water_tasks = addMaintenance(trees,"maple","water");
assert (water_tasks == 2);
```

Each of these function calls returns the number of trees that match and are newly scheduled to receive this maintenance. Note: We do not add duplicate maintenance tasks if a specific tree is already scheduled for that task.

Finally, we can print the data:

```
std::sort(trees.begin(),trees.end(),byNumTasks);
for (unsigned int i = 0; i < trees.size(); i++) {
  trees[i].print();
}
```

Which results in the following output, ordered first by number of scheduled tasks, the species, then age (planting year).

```
ash (1990) - prune insecticide
maple (1995) - water prune
oak (1950) - fertilize prune
maple (2015) - water
maple (2008) - water
pine (2002) - prune
oak (2012) -
```

6.1 `Tree` Class Declaration [16 pts]

Write the class declaration or *header file* (`tree.h`) for the `Tree` object. Use `const` and call by reference where appropriate. Remember, you should only implement 1 line member functions in the *class declaration*.

sample solution: 13 line(s) of code

6.2 `Tree` Class Member Function Implementation [10 pts]

Now implement the longer ($>$ 1 line) member functions as they would appear in the `tree.cpp` file.

sample solution: 14 line(s) of code

6.3 `Tree` Class Non Member Function Implementation [16 pts]

Finally, implement the three non-member functions used in the sample code related to the `Tree` class.

sample solution: 30 line(s) of code

7 Who's Eligible Now? [20 pts]

```
Smith 3
  Elizabeth 15.1
  Zachary 8.4
  Molly 2.1
Jones 1 Robert 12.2
Thomas 2
  Sally 2.7 John 4.9
```

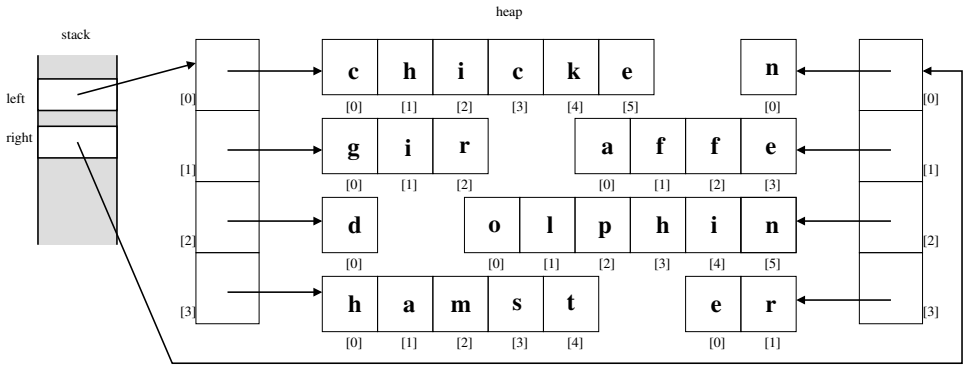
Anticipating the FDA's forthcoming approval Alyssa P. Hacker has been hired to prepare for the vaccination of children under 16 years of age. Help her write the `parse_children` function that takes in 3 arguments: the input filename of family data, the new minimum age for vaccination, and an initially-empty STL `vector` of `strings` to store the last names of families with at least one newly-eligible child (so we can send information about available vaccination appointments).

Each family in the file begins with the family's last name, then the number of children, and then each child's first name and floating point age. You should not use `getline` or `eof` to parse this file. The function returns the number of newly-eligible children (to buy sufficient doses). For this example, if the minimum age is 5.0, the `vector` should contain 2 `string`s (`Smith` and `Jones`) and the function should return 3.

sample solution: 21 line(s) of code

8 DynaMax Memory Cut [20]

We start with an STL `vector` of STL `strings` named `words`; for example: `chicken`, `giraffe`, `dolphin`, and `hamster`. Your task is to complete the code fragment below to construct the memory diagram on the right.



The diagram stores all of the characters from each word in arrays that are split or cut between the neighboring pair of letters that are furthest apart alphabetically.

`left =` ;

`right =` ;

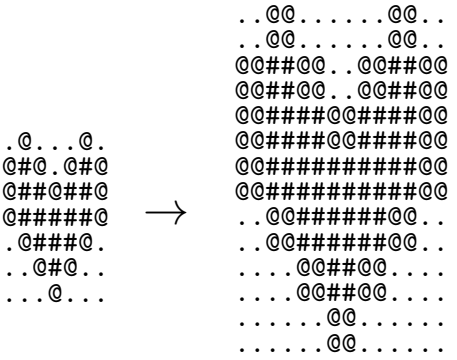
```
for (unsigned int w = 0; w < words.size(); w++) {
    unsigned int split = 0;
    int max = 0;
    for (unsigned int j = 0; j < words[w].size()-1; j++) {
        int diff = abs(words[w][j]-words[w][j+1]);
        if (diff > max) { split = j; max = diff; }
    }
```

sample solution: 9 line(s) of code

```
}
```

9 Image Doubler [/15]

Write a function named `doubler` that will double the dimensions (both width & height) of an ASCII art "image". Your function should take 2 arguments, named `input` and `output`, which are STL `vector`s of STL `string`s. An example is shown on the right.

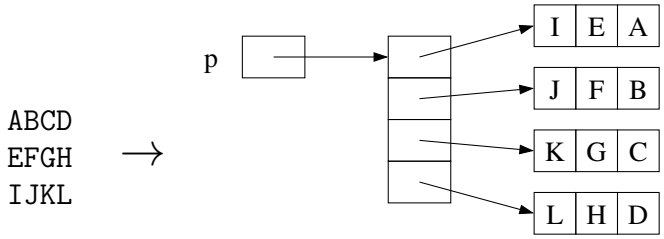


sample solution: 12 line(s) of code

What assumptions did you make about the data in `input`? What error checking should be done with the `input` variable before calling your function? Write 1-2 well-written and concise sentences.

10 Image Rotator [/20]

Now let's complete the fragment of code below to perform a 90° clockwise rotation of an ASCII art image. The input will again be an STL `vector` of STL `strings`. But this time, our output will be a two-dimensional dynamically allocated array, as shown in the diagram on the right.




```
// NOTE: we can assume input has been initialized with an interesting image
int rows = [                                                    ];
int cols = [                                                    ];
[                                                  ] p;
// dynamically allocate the structure and rotate the input
[                                                               ]
                                                  sample solution: 7 line(s) of code
// print the contents of p to the screen to help with debugging
for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
        std::cout << [                                          ];
    }
    std::cout << std::endl;
}
```

What is an alternate, equivalent expression for the previous box? *Hint: Use pointer arithmetic.*

[ ]

11 Searching for Substrings [/27]

Let's write a program to find all words containing a specific substring and print those words ordered by word length. Words of the same length will be in reverse alphabetical order. On this page you'll write two helper functions. *Read both pages of this problem before beginning your implementation.*

First, complete the definition of the `contains` helper function. Please write this "from scratch". Even if you know how to use STL's `substr` function or the C `char* strstr` function, don't use them.

```
bool contains(                                  ,                                  ) {
```

sample solution: 13 line(s) of code

Next, write another helper function:

sample solution: 4 line(s) of code

The program should expect 2 command line arguments: the filename for a dictionary containing lots of words and the target substring. For example, we might call this program:

```
./search_for_substring.out  dictionary.txt  con
```

Resulting in the output shown on the right. Note that for a full English dictionary there are over one thousand words containing the string "con". Complete the program below, using the helper functions you defined on the previous page.

```
icon
bacon
second
convoy
convex
economy
coconut
abscond
uncontrolled
inconsistent
```

```
int main(int argc, char* argv[]) {
  if (                                                                    ) {
    std::cerr << "Wrong number of arguments" << std::endl; exit(1);
  }

  if (                                                                    ) {
    std::cerr << "Cannot open file" << std::endl; exit(1);
  }

                                                    sample solution: 9 line(s) of code

  // print the words
  for (int i = 0; i < matches.size(); i++) {
    std::cout << matches[i] << std::endl;
  }
}
```

12 Classy Winners [/20]

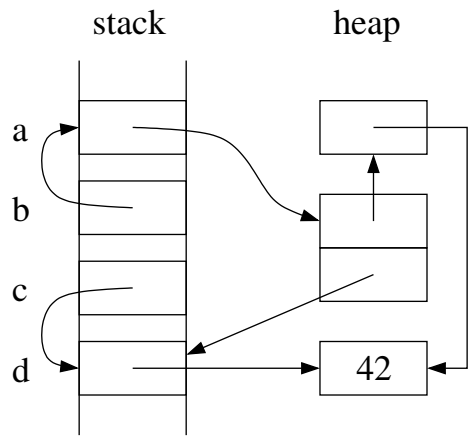
Write the two *header only* classes (they don't require a `.cpp` implementation file) used by the fragment of code on the right. You may omit `#include` and other pre-processor directives for your classes. This example code will print the following message to the screen: `The high jump winner is Mariya`

```
Winner hj("high jump");
hj.add(Finisher("Alessia",1.93));
hj.add(Finisher("Mariya",2.01));
hj.add(Finisher("Mirela",1.89));
hj.add(Finisher("Morgan",1.93));
hj.add(Finisher("Vashti",1.93));
hj.add(Finisher("Yulia",1.89));
hj.print();
```

sample solution: 9 line(s) of code

sample solution: 9 line(s) of code

13 Ben, Clean Up Your Mess! [/15]



Ben Bitdiddle left some messy code behind at the end of his summer internship (he probably won't be getting a return offer for next summer). The memory diagram for a key piece of the project (which is used many times, in many places) is shown on the right.

First, figure out the types for each variable. Complete this fragment of code to declare the 4 variables:

```
                                    a;
                                    b;
                                    c;
                                    d;
```

We skip the code where Ben dynamically-allocates and initializes the memory to make the picture above.

Finally, write a fragment of code to clean up the dynamically-allocated memory to ensure we don't have any memory leaks. Use at least 3 different variables.

sample solution: ? line(s) of code

What might happen to clients if they use this software without the cleanup code above? Write 1-2 well-written and concise sentences.