1 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 vectors of STL strings. An example is shown on the right.

```
.@@.@@.@@.@@.@@.@@
..@##@##@..@##@##@..@##@##@..@##@##@..@##@##@..@##@##@..@##@##@..@##@##@
...@...@...@...@...@...@...@...@...@...@...@...@...@...@...@...@...@...
```

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.
Now let's complete the fragment of code below to perform a $90^\circ$ 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.

```cpp
// 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

// print the contents of p to the screen to help with debugging

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

```cpp
bool contains( , ) {

sample solution: 13 line(s) of code
```

Next, write another helper function:

```cpp

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.

```cpp
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);
    }

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

*sample solution: 9 line(s) of code*
4  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

```cpp
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
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:

```c
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.
6 My Dog Ate My Budget [ /42]

Alyssa P. Hacker is helping her friend Ben Bitdiddle manage expenses so he can survive on a meager graduate TA stipend. IMPORTANT: Read through all 4 pages of this problem before beginning your implementation. Here’s an example main.cpp showing how Ben will record and manage his expenses:

```cpp
#include "budget.h"
int main() {
    Budget my_budget;
    my_budget.addExpense("home",Expense("water",20));
    my_budget.addExpense("food",Expense("groceries",100));
    my_budget.addExpense("home",Expense("rent",900));
    my_budget.addExpense("other",Expense("movie tickets",14));
    my_budget.addExpense("home",Expense("electricity",93));
    my_budget.addExpense("food",Expense("pizza",7));
    my_budget.print();
}
```

And here’s the nicely formatted output that will be printed when the program is run:
Note how the categories are alphabetized and the costs are sorted.

```
category: food     $ 107
    groceries     $ 100
    pizza         $  7
category: home     $ 1013
    rent          $ 900
    electricity   $  93
    water         $  20
category: other    $ 14
    movie tickets  $ 14
TOTAL              $ 1134
```

Before she had to leave, Alyssa wrote out the whole program on the whiteboard. After she left, Ben’s overly friendly (and very fluffy) dog, Muffit, erased a bunch of the code. Help Ben by filling in the blanks.

6.1 Expense Class Declaration (expense.h) [ /8]

Sample solution: 11 line(s) of code
# 6.2 Expense Class Implementation (expense.cpp) [2]

```cpp
#include "expense.h"

void Expense::print() const {
  std::cout << " " << std::left << std::setw(18) << name << "$
  << std::right << std::setw(5) << cost << std::endl;
}

byCost( ) {
  return (a.getCost()>b.getCost()) || (a.getCost()==b.getCost() && a.getName()<b.getName());
}
```

# 6.3 BudgetCategory Class Declaration (budget_category.h) [10]

```cpp
#include "expense.h"
```
6.4 BudgetCategory Class Implementation (budget_category.cpp) [ /5]

#include "budget_category.h"

void BudgetCategory::print() const {
    std::cout << "category: " << std::left << std::setw(10) << name << "\$" << std::right << std::setw(5) << getTotal() << std::endl;
    for (int i = 0; i < data.size(); i++) {
        data[i].print();
    }
}

addExpense( ) {

sample solution: 3 line(s) of code
}

byName( ) {
    return a.getName() < b.getName();
}

6.5 Budget Class Declaration (budget.h) [ /5]

#include "budget_category.h"

sample solution: 7 line(s) of code
6.6 Budget Class Implementation (budget.cpp)

```cpp
#include "budget.h"

void Budget::print() const {
    int total = 0;
    for (int i = 0; i < categories.size(); i++) {
        categories[i].print();
        total += categories[i].getTotal();
    }
    std::cout << std::left << std::setw(20) << "TOTAL" << "$" << std::right << std::setw(5) << total << std::endl;
}

addExpense( ) { }
```

6.7 Budget Compilation

Assuming the current directory contains these 7 files, what UNIX/GNU Linux command(s) would you type to compile and run this program?
Write a function named `suggest_words_vertical` that takes four arguments: a partially completed crossword puzzle board and a dictionary of words (both represented as vectors of strings), and two integers that specify a row and column location in the board. Similarly to Homework 1, the board will use the '#' character to represent a black square. The other squares will either have a lowercase letter 'a'-'z' or a '?' character which indicates the square needs to be filled in. Your function should return a vector of all legal words from the dictionary that could fit as the vertical word starting at the specified row and column.

Note: Your function doesn’t need to worry about the horizontal words or error checking.

For example, given the board above, the call `suggest_words_vertical(board, dictionary, 1, 1)` returns `tide tike tile time tine tire` and `suggest_words_vertical(board, dictionary, 0, 3)` returns `shoe shop shot show slob slog slot slow smog snob snow soon soot spot stop stow`.

Sample solution: 20 line(s) of code
For each of the memory diagrams below, write a fragment of code that will produce the diagram.

**sample solution: 10 line(s) of code**

```
stack    heap
stairs:
```

```
stack    heap
zig:
zag:
```

```
stack    heap
cumulus:
cirrus:
```

```
stack    heap
fluffy: true
height: 3.14
```

```
stack    heap
fluffy: false
height: 6.02
```

```
sample solution: 4 line(s) of code
```

```
sample solution: 9 line(s) of code
```
Complete the code fragment below to implement a simplified notification system for a messaging platform similar to Twitter. The input file is formatted as multiple messages. Each message consists of the sender, the size of the message (number of words), and the words of the message. A sample of the data is shown above. Note that messages may wrap over multiple lines. After reading each message you will send notifications to everyone tagged with an '@' in the message body. Here is the expected output:

Sally->Fred: 'I just ate lunch with @Fred at Bella's and it was delicious.'
Fred->Sally: 'Mmmm... food... good! @Sally @Joe'
Fred->Joe: 'Mmmm... food... good! @Sally @Joe'
Joe->Fred: 'Hey @Fred why didn't you invite me?'
Fred->Sally: '@Sally next time let's call Joe!'
In this problem we will search a large ASCII Art canvas for matches to a target pattern. For example, given the ASCII Art Tic Tac Toe game on the right, we can search for the 'X' symbol on the board. Or the 'O' symbol. Or the height=5 column of '|' characters that make the vertical bar of the Tic Tac Toe board. You will write a function named `search_for_symbols` that takes in 2 arguments, `canvas` and `symbol`. Both inputs are STL vectors of STL strings. The function should return all locations of that pattern or symbol on the canvas. The return type will be an STL vector of positions where each position is an STL vector of integers. Note: Your function program should be general, and work with any size canvas and any size symbol.

In the example below we search for all of the 'O' symbols on the board:

```c++
std::vector<std::string> symbol;
symbol.push_back(" *** ");
symbol.push_back("* *");
symbol.push_back("* *");
symbol.push_back("* *");
symbol.push_back(" *** ");
std::vector<std::vector<int> > positions = search_for_symbols(tic_tac_toe_canvas,symbol);
for (int i = 0; i < positions.size(); i++) {
    assert (positions[i].size() == 2);
    std::cout << "An O is located at: " << positions[i][0] << "," << positions[i][1] << std::endl;
}
```

And this is the expected output:

```
An O is located at: 0,6
An O is located at: 12,12
```

10.1 Assumptions & Error Checking & Realistic Expectations

You will implement the `search_for_symbols` function on the next page. You will not need to do any error checking in your function. But let’s pause and consider what assumptions you are making about the input and output. Write 3-4 well-written and concise sentences. You may find it easier to write the function on the next page, then return to answer this part.
10.2 Now Implement the search_for_symbols Function

sample solution: 22 line(s) of code
Stack Sleuthing [ /15]

Carefully examine the contents of the stack after program execution shown on the right.
Your task for this problem is to complete the source code on the left so that it produces this stack.

```cpp
11 Stack Sleuthing [ /15]

Carefully examine the contents of the stack after program execution shown on the right.
Your task for this problem is to complete the source code on the left so that it produces this stack.

function main

<table>
<thead>
<tr>
<th>x:</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>y:</td>
<td>3</td>
</tr>
<tr>
<td>tmp:</td>
<td>42</td>
</tr>
<tr>
<td>b:</td>
<td>3</td>
</tr>
<tr>
<td>a:</td>
<td>8</td>
</tr>
<tr>
<td>return address:</td>
<td>main@line13</td>
</tr>
</tbody>
</table>

function foo

<table>
<thead>
<tr>
<th>answer:</th>
<th>42</th>
</tr>
</thead>
<tbody>
<tr>
<td>b:</td>
<td>5</td>
</tr>
<tr>
<td>a:</td>
<td>7</td>
</tr>
<tr>
<td>return address:</td>
<td>foo@line6</td>
</tr>
</tbody>
</table>

function foo

<table>
<thead>
<tr>
<th>answer:</th>
<th>42</th>
</tr>
</thead>
<tbody>
<tr>
<td>b:</td>
<td>7</td>
</tr>
<tr>
<td>a:</td>
<td>6</td>
</tr>
<tr>
<td>return address:</td>
<td>foo@line6</td>
</tr>
</tbody>
</table>

function foo

| answer: | 42 |
```

```cpp
1 { }  
2 }  
3 if (  
4 }  
5 else  
6 }  
7 }  
8 }  
9 }  
10 }  
11 }  
12 }  
13 }  
14 std::cout << tmp << std::endl;  
15 }
```
For each code fragment below, choose the letter that best describes the program error. *Hint: Each letter will be used exactly once.*

A ) Accessing data beyond the array bounds
B ) Uninitialized memory
C ) Infinite loop
D ) Compile error: type mismatch
E ) Does not contain an error
F ) Memory leak
G ) Math error (incorrect answer)
H ) Syntax error

```c
int* apple;
int banana[5] = {1, 2, 3, 4, 5};
apple = &banana[2];
*apple = 6;

std::vector<std::string> temperature;
temperature.push_back(43.5);
```

```c
float* floating_pt_ptr = new float;
*floating_pt_ptr = 5.3;
floating_pt_ptr = NULL;
```

```c
float a = 2.0;
float b = -11.0;
float c = 12.0;
float pos_root = -b + sqrt(b*b - 4*a*c) / 2*a;
float neg_root = -b - sqrt(b*b - 4*a*c) / 2*a;
```

```c
for (int i = 0; i < 10; i++) {
x += sqrt(double(i));
}
```

```c
std::vector<std::string> temperature;
temperature.push_back(43.5);
```

```c
unsigned int x;
for (x = 10; x >= 0; x--) {
    std::cout << x << std::endl;
}
```

```c
int balance = 100;
int withdrawal;
std::cin >> withdrawal;
if (withdrawal <= balance)
    balance -= withdrawal;
else
    std::cout << "failure\n";
```

```c
int grades[4] = { 1, 2, 3, 4 };
std::cout << "grades" << grades[1]
    " " << grades[2]
    " " << grades[3]
    " " << grades[4]
    " " << std::endl;
```
In this problem you will implement a simple class named `RunTracker` to keep track of an athlete’s running workouts. **IMPORTANT: Read through all 3 pages of this problem before beginning your portion of the implementation.**

First we create a number of runners, and store their recent runs by distance (in miles):

```cpp
RunTracker george("George", "Smith");
RunTracker sally("Sally", "Williams");
RunTracker chris("Chris", "Jones");
george.addRun(6.7); sally.addRun(3.1); chris.addRun(2.5); george.addRun(5);
sally.addRun(13.1); chris.addRun(2); george.addRun(9.6); chris.addRun(26.2);
sally.addRun(13.1); george.addRun(11.7);
```

```cpp
std::vector<RunTracker> runners;
runners.push_back(george); runners.push_back(sally); runners.push_back(chris);
```

We’d like to count how many times each runner ran a specific distance (or more). For example:

```cpp
std::cout << "Runners who have completed the half marathon distance: " << std::endl;
for (unsigned int i = 0; i < runners.size(); i++) {
    int num = runners[i].numRunsAtLeast(13.1);
    if (num >= 1)
      std::cout << " " << runners[i].getName() << " (" << num << ")" << std::endl;
}
```

Should result in this output:

Runners who have completed the half marathon distance:
  Williams, Sally (2)
  Jones, Chris (1)

Finally, we’d like to print all runners, ordered by total distance completed. The following code:

```cpp
/*/ ONE LINE OF CODE OMITTED: YOU WILL FILL IN THIS LINE IN PART 1 */

for (unsigned int i = 0; i < runners.size(); i++) {
    std::cout << std::setw(18) << std::left << runners[i].getName() << " " << std::setw(4) << std::right << std::fixed << std::setprecision(1) << runners[i].totalDistance() << std::endl;
}
```

Should result in the following output:

Williams, Sally   29.3
Jones, Chris     30.7
Smith, George    33.0

### 13.1 Omitted Line of Code

What is the missing line of code above? It may depend on your answers on the next page(s).
13.2 RunTracker Class Declaration [ /14]

Using the sample code on the previous pages as your guide, write the class declaration for the RunTracker object. That is, write the header file (run_tracker.h) for this class. You don’t need to worry about the #include lines or other pre-processor directives. Focus on getting the member variable types and member and non-member function prototypes correct. Use const and call by reference where appropriate. Make sure you label what parts of the class are public and private. Save the implementation of all functions for the run_tracker.cpp file, which is the next part.

sample solution: 13 line(s) of code
Now implement all of the functions prototyped in the `run_tracker.h` file, as they would appear in the corresponding `run_tracker.cpp` file.
The code below is missing its variable declaration & initialization. Fill in these lines so the code compiles and runs without syntax errors, memory errors or leaks. Show your work by diagramming the memory.

```c
int main() {
    c[2][0] = 'R';
    (*a)[0] = NULL;
    (*b)[2][1] = 'P';
    c[1] = NULL;
    *(d[1]) = 'I';
    d[0] = NULL;

    // CREATE
    // MEMORY
    // DIAGRAM

    delete [] (*a)[2];
    delete b;
    delete [] c;
    delete d[1];
}
```