1. Random conceptual questions
   a. What does the → operator do?
   b. What do the two keywords that allow us to dynamically allocate memory do?
   c. What does the :: operator do?
   d. What's the difference between → and (*...). ?
   e. What does a pointer variable hold?
   f. How do you dereference a pointer?

2.

```
int* x = 0;
int* p = &x;
*p = 4
x = 5 + *p
cout << *p << endl;
```

What does the following code output?

3.

```
int arr[] = [0, 5, 8, 23, 97, 2, 67, 1]
int* num_ptr = arr;
```

Write *(arr + 2) in array notation using num_ptr. Write the value too.

4.
Write a recursive function that returns the number of digits in an integer.

```
int num_digits(int n) {
```

5.
Write a function that returns a dynamic array of size n that contains [1.0,0.5,0.25,...] or in other words containing 1.0 in the 0th index and half of the previous value in each of the following indexes.

```
float* decreasingFloats(int n) {
```

6. Write a function that checks if char c or char a is in string s.

```
bool stringChecker(string& s, char a, char b) {
```

7. Write a function that delete int n from a sorted vector filled with unique integers. The vector must maintain its order.

```
void deleteNum(vector<int>& myVector, int n) {
```

8.
a. Write a function that returns a dynamic array of the averages of each row in a two-dimensional array with int number_of_rows rows and int COLUMNS columns.

```
const int COLUMNS = 4;
int* averageArray(int arr[][COLUMNS], int number_of_rows) {
```

b. Free up allocated memory in the main function and handle the dangling pointer.

```
int main() {
   // Example usage
   const int number_of_rows = 3;
   int myArray[][COLUMNS] = {{1, 2, 3, 4},
                             {5, 6, 7, 8},
                             {9, 10, 11, 12}};

   // Call the function and get the array of averages
   int* result = averageArray(myArray, number_of_rows);

   // Print the array of averages
   for (int i = 0; i < number_of_rows; ++i) {
      std::cout << "Average for row " << i + 1 << ": " << result[i] << std::endl;
   }
```

9.
Write code that fills an array of pointers with dynamically allocated arrays to create a triangular array filled with 0’s. When printed out, it should look like this:

```
X X X X
X X X
X X
X
```

```
const int ROWS = 4;
int* counts[ROWS];

// allocate memory for arrays and fill 0’s
```

```
// print out counts to look like the triangular array above
```

```
// deallocate memory and handle dangling pointers
```

10.

a. Implement a class ShoppingList that uses a string vector. Implement a constructor that constructs a ShoppingList with an empty string vector. Implement the following member functions aswell: int get_number_of_items (whcih returns the number of items in the shopping list), void add_item (which adds an item to the shopping list), and void remove_item (which removes an the most recently added item from the shopping list).

b. Write code that does the following:

1) Dynamically construct a ShoppingList.

2) Add milk and bread.

3) Print out how many items are in the shopping list. What is that number?

4) Remove an item. Print out how many items are in the shopping list. What is that number now?

5) We will no longer have to use this ShoppingList. Free up allocated memory and handle the dangling pointer.