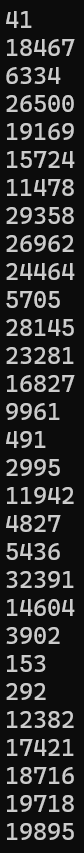
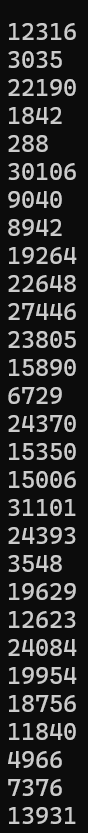
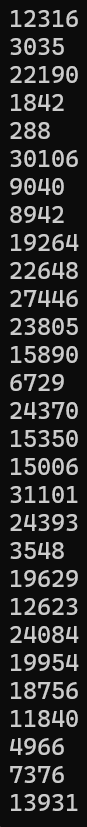
1. Example Code

|  |
| --- |
| void problemOne() {  srand(50);  int nums[50];  for (int i = 0; i < 50; i++) {  nums[i] = 100 + (rand() % 101);  std::cout << nums[i] << " ";  }  } |

What its doing: Setting the seed for random to 50 then generating 50 random numbers between 100 to 200. Because the seed isn’t changed between runs the numbers will be the same for each run.

1. Learning about rand() and srand()

|  |
| --- |
| void problemTwo() {  for (int i = 0; i < 30; i++) {  std::cout << rand() << std::endl;  }  } |

Output:  
 

1. Pseudo-random integer numbers

|  |
| --- |
| void problemThree() {  srand(3);  int nums[50];  for (int i = 0; i < 50; i++) {  nums[i] = rand() % 101;  }  int size = sizeof(nums) / sizeof(int);  // Sorting  for (int i = 0; i < size - 1; ++i) {  for (int j = 0; j < size - i - 1; ++j) {  if (nums[j] > nums[j + 1]) {  int temp = nums[j];  nums[j] = nums[j + 1];  nums[j + 1] = temp;  }  }  }    // Outputting Orginal Array  std::cout << "Original Array:\n";  for (int i = 0; i < size; i++) {  std::cout << nums[i] << " ";  }  std::cout << std::endl;  // Getting user input for k  int k = 0;  std::cout << "For what number k?: ";  std::cin >> k;  std::cout << "Deleting duplicates for the first " << k << " number unique elements\n";  // Deleting duplicates up to k elements  int newSize = size;  for (int i = 0; i < k; ++i) {  for (int j = i + 1; j < newSize; ++j) {  if (nums[i] == nums[j]) {  for (int z = j; z < newSize - 1; ++z) {  nums[z] = nums[z + 1];  }  --newSize;  --j;  }  }  }  std::cout << "New Array:\n";  for (int i = 0; i < newSize; i++) {  std::cout << nums[i] << " ";  }  std::cout << std::endl;  } |

Output:  
A black background with white text

Description automatically generated

1. Recursion 1

|  |
| --- |
| void problemFour() {  int b, e;  std::cout << "Input the base number and the exponent" << std::endl;  std::cin >> b >> e;  std::cout << "Result = " << power(b, e) <<std::endl;  }  int power(int b, int e) {  if (e == 0) {  return 1;  }  return b \* power(b, e - 1);  } |

Output:

A black background with white text

Description automatically generated

1. Recursion 2

|  |
| --- |
| void problemFive() {  std::string input;  std::cout << "Please input a string: ";  std::cin >> input;  std::cout << (palindrome(input,0,input.length()-1) ? "Palindrome!" : "Not a palindrome...");  }  bool palindrome(std::string s, int start, int end) {  if (start >= end) {  return true;  }  else if (s[start] == s[end]) {  palindrome(s, ++start, --end);  }  else  return false;  } |

Output:  
A black background with white text

Description automatically generated

A black background with white text

Description automatically generated

1. Reverse array elements

|  |
| --- |
| void problemSix(){  int nums[] = { 1,2,3,4,5,6,7,8,9 };  size\_t size = sizeof(nums) / sizeof(int);    std::cout << "Original Array:\n";  for (int i = 0; i < size; i++) {  std::cout << nums[i] << " ";  }  std::cout << std::endl;  reverse\_array(nums, size);  std::cout << "Reversed Array:\n";  for (int i = 0; i < size; i++) {  std::cout << nums[i] << " ";  }  std::cout << std::endl;  }  void reverse\_array(int array[], size\_t arraySize) {  int start = 0, end = arraySize - 1;  int temp = 0;  while (start < end) {  temp = array[start];  array[start] = array[end];  array[end] = temp;  start++;  end--;  }  } |

Output:  
A black background with white text

Description automatically generated