Steven Chu

905-094-800

CS 31 Project 6

a)

#include <iostream>

**using** **namespace** std;

**int** main()

{

**int** arr[4] = { 0, 1, 2, 3 };

**int**\* ptr = arr;

\*ptr = arr[ 1 ]; // set arr[0] to 1

\*(ptr + 1) = arr[ 0 ] \* 10; // set arr[1] to 10

ptr += 2;

\*ptr = arr[ 1 ] \* 10; // set arr[2] to 100

\*(ptr + 1) = 1000; // set arr[3] to 1000

ptr = arr;

**while** (ptr <= &arr[3]) // loop over the whole array

{

cout << " " << \*ptr; // print a single value

ptr++;

}

cout << endl;

**return**( 0 );

}

b) This function returns the address of the last zero and not the first zero because of the way the for loop was constructed.

**void** findFirstZero(**int** arr[], **int** n, **int**\* p)

{

p = **nullptr**; /// default value if there isn't a 0 in the array at all

**for** (**int** k = 0; k < n; k++)

{

**if** (arr[k] == 0) // found an element whose value is 0

// since we want the first zero value, as we keeping looping, we'll get to the 0 that is behind us in the array...

{

p = arr + k; // change the value of p

}

}

}

c) The main may not work because the pointer variable is not initialized to a proper memory value before being used as a parameter.

**int** main()

{

**int** i;

**int**\* p = &i;

smallest(15, 20, p);

cout << "The smallest value is " << \*p << endl;

**return**( 0 );

}

d) Match function used wrong syntax to indicate null terminator, attempted to increment an array pointer, which is static, and attempted to compare pointer variables without dereferencing them.

**bool** match(**const** **char** str1[], **const** **char** str2[])

{

**int** i = 0;

**int** j = 0;

**bool** result = **true**;

**while** (\*(str1+i) != '\0' && \*(str2+j) != '\0') // zero bytes at ends

{

**if** (\*(str1+i) != \*(str2+j)) // compare corresponding characters

{

result = **false**;

**break**;

}

i++; // advance to the next character

j++;

}

**if** (result)

{

result = (i == j); // both ended at same time?

}

**return**( result );

}

e) This program will not work because the function computeFibonacciSequence() is returning a pointer address of arr that is deleted after the function passes out of scope. Therefore trying to use this address in the main routine will result in random values being printed.

1. This program prints the reverse of the original array.

#include <iostream>

**using** **namespace** std;

**int** main( )

{

**int** x[ 5 ] = { 1, 2, 3, 4, 5 }; //initializes array of 5 elements

**int** \* p = x; //creates a pointer variable pointing to x[0]

**int** i; //initializes an int

**for** (i = 0; i < 2; i++)

{

**int** temp = \*(p + i); //creates a holder for the int value at x[i]

\*(p + i) = \*(p + 4 - i); //reflects the value of x[4-i] across the middle value

\*(p +4 - i) = temp; //sets x[4-i] to the original value of x[i]

}

**for** (i = 0; i < 5; i++)

{

cout << x[i] << " "; //prints each value of the array

}

cout << endl;

**return**( 0 );

}

1. The program prints “5.999 5.999 5.9995.999”. The second \* is necessary to make sure z is also a pointer variable and not just a regular double.

#include <iostream>

**using** **namespace** std;

**int** main()

{

**double** x = 5.999; //creates a double value

**double** \*y, \*z; // Line A - why is the second \* required?

//creates two pointer variables

y = &x; //sets pointer y to point at the address of x

z = y; //sets pointer z to also point to the address of x

cout << x << " " << \*(&x) << " " << \*y << \*z << endl;

//prints: "5.999 5.999 5.9995.999"

**return**( 0 );

}

1. This program prints “10 50 40 0”

#include <iostream>

**using** **namespace** std;

**int** main()

{

**int** track[ ] = { 10, 20, 30, 40 }; //initializes an array of 4 numbers

**int** \* ptr; //creates an int pointer variable

ptr = track; //sets ptr to point at track[0]

track[1] += 30; //sets track[1] to 50

cout << \* ptr << " "; //prints track[0] (10)

\*ptr -= 10; //sets track[0] to 0

ptr++; //increments ptr to point at track[1]

cout << \* ptr << " "; //prints track[1] (50)

ptr += 2; //increments ptr to point at track[3]

cout << \* ptr << " "; //prints track[3] (40)

cout << track[ 0 ] << endl; //prints new value track[0] (0)

**return**( 0 );

}

#include <iostream>

**using** **namespace** std;

**int** main()

{

**int** num[ 5 ];

// Line A

// rewrite all this code without this declared variable above

// make your revised code do the same thing as this original logic

**int** counter = 0;

num[counter] = 100;

counter++;

num[counter] = 90;

counter = 2;

num[counter] = 80;

counter = 0 + 3;

num[counter] = 70;

counter = 0;

num[counter + 4] = 60;

**for** (**int** i = 0; i < 5; i++)

cout << num[ i ] << " ";

cout << endl;

**return**( 0 );

}

#include <iostream>

**using** **namespace** std;

**void** deleteDigits(**char**\* message)

{

**char**\* placeholder = message;

**while**(\*(message) != '\0')

{

**if**(\*(message) == '0' || \*(message) == '1' || \*(message) == '2' || \*(message) == '3' || \*(message) == '4' || \*(message) == '5' || \*(message) == '6' || \*(message) == '7' || \*(message) == '8' || \*(message) == '9')

{

**while**(\*(message) != '\0')

{

\*(message) = \*(message+1);

message++;

}

message = placeholder;

}

message++;

}

}

**int** main()

{

**char** msg[100] = "Happy 2019!";

deleteDigits(msg);

cout << msg << endl; // prints: Happy !

}