Amy Yu

905138432

Project 6

1. a.

int main()

{

int arr[3] = { 5, 10, 15 };

int\* ptr = arr;

\*ptr = 10; // set arr[0] to 10

\*(ptr + 1) = 20; // set arr[1] to 20

ptr += 2;

ptr[0] = 30; // set arr[2] to 30

while (ptr >= arr)

{

cout << ' ' << \*ptr; // print values

ptr--;

}

cout << endl;

}

b. In the findDisorder function, the pointer must be passed by reference using the ampersand because it is being modified in the main function.

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

{

for (int k = 1; k < n; k++)

{

if (arr[k] < arr[k-1])

{

p = arr + k;

return;

}

}

p = nullptr;

}

c. The pointer in the main function is not initialized.

int main()

{

double n = 0;

double\* p = &n;

hypotenuse(1.5, 2.0, p);

cout << "The hypotenuse is " << \*p << endl;

}

d. The function is comparing the addresses of the characters instead of the actual character, which will always be false. The asterisk should be used to create a pointer to the character in the C string. The while loop was also checking if the characters are equal to 0 instead of a null byte, which should be ‘\0’.

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

{

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

{

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

return false;

str1++; // advance to the next character

str2++;

}

return (\*str1 == \*str2); // both ended at same time?

}

int main()

{

char a[10] = "pointy";

char b[10] = "pointless";

if (match(a,b))

cout << "They're the same!\n";

}

e. The array only exists inside the function, so it cannot be accessed in the main function.

1. a. string\* fp;

b. string fish[5];

c. fp = fish + 4;

d. \*fp = “yellowtail”;

e. \*(fish + 3) = “salmon”;

f. fp -= 3;

g. fp[1] = “sole”;

h. fp[0] = “eel”;

i. bool d = (fp == fish);

j. bool b = (\*fp == \*(fp+1));

1. a.

double computeAverage(const double\* scores, int nScores)

{

int n = 0;

double tot = 0;

while (n!= nScores)

{

tot += (scores+n);

n++;

}

return tot/nScores;

}

b.

const char\* findTheChar(const char\* str, char chr)

{

for (int k = 0; \*(str + k) != 0; k++)

if (\*(str + k ) == chr)

return (str + k);

return nullptr;

}

c.

const char\* findTheChar(const char\* str, char chr)

{

while (\*str != ‘\0’)

{

if (\*str == chr)

return str;

else

str++;

}

return nullptr;

}

#include <iostream>

using namespace std;

int\* minimart(int\* a, int\* b)

{

// returns pointer to a if the value of what a is pointing to is less than the value of what b is pointing to

if (\*a < \*b)

return a;

// otherwise returns b

// returns pointer to array[2] since array[0] is not less than array[2]

else

return b;

}

void swap1(int\* a, int \*b) // swaps two pointers

{

int\* temp = a;

a = b;

b = temp;

}

void swap2(int\* a, int \*b) // swaps the values of the pointers

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main()

{

int array[6] = { 5, 3, 4, 17, 22, 19 };

int\* ptr = minimart(array, &array[2]); // ptr is pointing to array[2]

ptr[1] = 9; // sets array[3] to 9

ptr += 2; // ptr points to array[4]

\*ptr = -1; // sets array[4] to -1

\*(array+1) = 79; // sets array[1] to 79

cout << "diff=" << &array[5] - ptr << endl; // prints diff=1 because ptr is the memory address of array[4], so &array[5]-&array[4] is 1

swap1(&array[0], &array[1]); // swaps the memory addresses of array[0] and array[1]

swap2(array, &array[2]); // swaps the values of array[0] and array[2]

for (int i = 0; i < 6; i++)

cout << array[i] << endl; // prints all the values of the array on a new line

}

This program prints:

diff=1

4

79

5

9

-1

19

void deleteG(char\* phrase)

{

char\* newPhrase = phrase;

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

{

if (\*newPhrase != 'g' && \*newPhrase != 'G')

{

\*phrase = \*newPhrase;

newPhrase++;

phrase++;

}

else

newPhrase++;

}

}