Project 6

1a. int main()

{

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

int\* ptr = arr;

\*ptr = 30;

\*ptr + 1 = 20; // error

ptr += 2;

ptr[0] = 10; // error

while (ptr >= arr)

{

ptr--; // wrong place

cout << \*ptr << endl;

}

}

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 << endl; // print values

ptr--;

}

}

1b. The problem is that the pointer is not passed by reference and is just returning a random pointer and not changing the pointer that the main calls for.

void findMax(int arr[], int n, int\* &pToMax)

{

if (n <= 0)

return; // no items, no maximum!

pToMax = arr;

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

{

if (arr[i] > \*pToMax)

pToMax = arr + i;

}

}

int main()

{

int nums[4] = { 5, 3, 15, 6 };

int\* ptr;

findMax(nums, 4, ptr);

cout << "The maximum is at address " << ptr << endl;

cout << "It's at position " << ptr - nums << endl;

cout << "Its value is " << \*ptr << endl;

}

1c. The pointer variable ptr is not initialized to anything, so it is not pointing to anything. To fix this, we can set a temporary variable to the pointer.

void computeCube(int n, int\* ncubed)

{

\*ncubed = n \* n \* n;

}

int main()

{

int number = 0;

int\* ptr = &number;

computeCube(5, ptr);

cout << "Five cubed is " << \*ptr << endl;

}

1d. The problem is that the program is trying to compare C strings directly as if we were only comparing integer variables, and it is not using the correct syntax for comparing C strings without pointers. To fix this while using this syntax, we are comparing the values that the pointer points to instead of comparing them directly. It loops through the array while comparing the value of the pointers until the pointer reaches the zero byte.

bool strequal(const char\* str1[], const char\* str2[])

{

while (\*str1 != ‘\0’ && \*str2 != ‘\0’)

{

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

return false;

str1++; // advance to the next character

str2++;

}

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

}

1e. In the function getPtrToArray, it creates a local array called anArray that does not carry on through the program. Once the function is called and finished, the previous elements in anArray are not kept and are set to undefined values. Any further attempts to change the array just displays undefined values.

2.

a. double\* cat;

b. double mouse[5];

c. cat = mouse + 4;

d. \*cat = 25;

e. \*(mouse + 3) = 42;

f. cat -= 3;

g. cat[1] = 27;

h. cat[0] = 54;

i. bool b = (\*cat == \*(cat + 1));

j. bool d = (cat == &mouse[0]);

3a. double mean(const double\* scores, int numScores)

{

double tot = 0;

for (int x = 0; x < numScores; x++)

tot += \*(scores + x);

return tot/numScores;

}

3b. const char\* findTheChar(char\* str, char chr)

{

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

{

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

return (str + k);

}

return nullptr;

}

3c. const char\* findTheChar(char\* str, char chr)

{

while (\*str != ‘\0’)

{

if (\*str == chr)

return str;

str++;

}

return nullptr;

}

4. int\* maxwell(int\* a, int\* b) //creates a pointer to larger reference

{

if (\*a > \*b)

return a;

else

return b;

}

void swap1(int\* a, int\* b) //swap the pointers in the parameter

{

int\* temp = a;

a = b;

b = temp;

}

void swap2(int\* a, int\* b) //swaps what the parameters point to

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main()

{

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

int\* ptr = maxwell(array, &array[2]); //sets pointer to array[0]

\*ptr = -1; //sets value of pointer to -1

ptr += 2; //moves ptr two places to the right to array[2]

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

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

cout << &array[5] - ptr << endl; //prints difference between array[5] and array[2], which is 3

swap1(&array[0], &array[1]); //swaps pointer of array[0] and array[1]

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

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

cout << array[i] << endl; //prints through array

}

Output is:

3

4

79

-1

9

22

19

5. void removeS(char\* message)

{  
 char\* msg = message;

while (\*msg != ‘\0’)

{

if (\*msg == ‘S’ || \*msg == ‘s’)

{  
 while (\*msg != ‘\0’)

{  
 \*msg = \*(msg + 1);

msg++;

}

msg = message;

}

msg++;

}

}