**Question 1a**

int main()

{

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

int\* ptr = arr;

\*ptr = 30; // set arr[0] to 30

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

ptr += 2;

ptr[0] = 10; // set arr[2] to 10

ptr = arr; // set ptr to 0

while (ptr <= (arr+2))

{

cout << \*ptr << endl;

ptr++;

}

}

**Question 1b**

In the original function, the value of pToMax was only changed within the function, and thus in the main function it was not changed. We have to make pToMax a call by reference.

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;

}

**Question 1c**

The pointer is uninitialized. Correct the code by initializing the pointer to an int x.

void computeCube(int n, int\* ncubed)

{

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

cout << \*ncubed << endl;

}

int main()

{

int x;

int\* ptr;

\*ptr = x;

computeCube(5, ptr);

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

}

**Question 1d**

Need to use pointers to refer to value at positions in str1 and str2.

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?

}

int main()

{

char a[15] = "Noor";

char b[15] = "Noor";

if (strequal(a,b))

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

}

**Question 1e**

The storage for the local variable anArray goes away when the function getPtrToArray returns because the program has left the function and is in the main routine. As the function returns a pointer to that storage, calling that pointer in the main routine using ptr[i]) results in undefined behavior.

**Question 2a**

double\* cat;

**Question 2b**

double mouse[5];

**Question 2c**

cat = &mouse[4]; OR

cat = mouse + 4;

**Question 2d**

\*cat = 42;

**Question 2e**

\*(mouse + 3) = 25;

**Question 2f**

cat -= 3;

**Question 2g**

cat[1] = 17;

**Question 2h**

cat[0] = 54;

**Question 2i**

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

**Question 2j**

bool d = (cat == mouse); OR

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

**Question 3a**

double mean(const double\* scores, int numScores)

{

int k = 0;

double tot = 0;

while (k != numScores)

{

tot += \*(scores + k);

k++;

}

return tot/numScores;

}

**Question 3b**

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;

}

**Question 3c**

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

{

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

if (\*str == chr)

return str;

return nullptr;

}

**Question 4**

3 (6.)

4 (7. 8.)

79 (5. 7.)

-1 (2. 8.)

9 (4.)

22 (doesn’t change)

19 (doesn’t change)

1. maxwell is called with pointers to array[0] and array[2]. It returns a pointer to whichever of the ints pointed to has a bigger value. Since array[0] has the bigger value, function returns array[0].
2. Expression \*ptr = -1 sets array[0] to -1.
3. ptr += 2; sets ptr to array[2].
4. ptr[1] = 9; sets array[3] to 9.
5. \*(array+1) = 79; sets array[1] to 79.
6. &array[5] – ptr is 3 because 5 - 2 is 3 (i.e. their memory locations are 3 spots apart).
7. swap1 function swaps its copies of the pointers passed in to it, with no effect on the ints pointed to.
8. swap2 function swaps the ints pointed to.

**Question 5**

void removeS(char\* source)

{

char\* destination = source;

for ( ; \*source != '\0'; source++)

{

if (\*source != 's' && \*source != 'S')

{

\*destination = \*source;

destination++;

}

}

\*destination = '\0';

}