Project 6

* 1. The first bug in this program is that \*ptr + 1 can’t be assigned to 20. The second bug is that the while loop should start from the beginning of the array and loop to the end, not from the end to the beginning; otherwise, the output will read “10 20 30” instead of “30 20 10.” The third error is that ptr should be incremented after printing out the array element. The fourth error is that ptr should be incremented up 1 instead of down 1. The function should be:  
       
     int main() {  
      int arr[3] = {5, 10, 15};  
      int\* ptr = arr;  
       
      \*ptr = 30;  
      \*(ptr + 1) = 20;  
      ptr += 2;  
      \*ptr = 10;  
       
      while (ptr <= arr+2) {  
      cout << \*ptr << endl;  
      ptr++;  
      }  
     }
  2. The function won’t set pToMax parameter to point to the maximum item because it remains uninitialized in the function. To fix this, the third parameter should be int\*& pToMax instead of int\* pToMax. By doing this, the pointer gets passed by reference, therefore directly changing it rather than making a copy. The function should be:  
       
     void findMax(int arr[], int n, int\* pToMax) {  
      if (n <= 0)  
      return;  
      pToMax = arr;  
      for (int i = 1; i < n; i++) {  
      if (arr[i] > \*pToMax)  
      pToMax = arr + i;  
      }  
     }
  3. It doesn’t work because ptr is uninitialized. A possible solution is declaring a variable to which ptr points. The main routine should be:  
       
     int main() {  
      int n;  
      int\* ptr = &n;  
      computeCube(5, ptr);  
      cout << “Five cubed is ” << \*ptr << endl;  
     }
  4. This method compares the C strings themselves instead of their pointers. Without the dereference operator \*, the if statement will always return false. The function should be:  
       
     bool strequal(const char str1[], const char str2[]) {  
      while (\*str1 != 0 && \*str2 != 0) {  
      if (\*str1 != \*str2)  
      return false;  
      str1++;  
      str2++;  
      }  
      return \*str1 == \*str2;  
     }
  5. The program produces unexpected output because anArray is contained within the scope of the getPtrToArray function. The main method receives the pointer returned by the function, but can’t access the array.
  6. double\* cat;
  7. double mouse[5];
  8. cat = mouse + 4
  9. \*cat = 25;
  10. \*(mouse + 3) = 54;
  11. cat -= 3
  12. cat[2] = 42;
  13. cat[] = 27;
  14. bool b = (\*cat == \*(cat+1));
  15. bool d = (\*cat == mouse)
  16. double mean(const double\* scores, int numScores) {  
       double tot = 0;  
       for (int i = 0; i < numScores; i++)  
       tot += \*(scores+i);  
       return tot/numScores;  
      }
  17. 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;  
      }
  18. const char\* findTheChar(const char str[], char chr) {  
       while (\*str != 0) {  
       if (\*str == chr)  
       return str;  
       str++;  
       }  
       return nullptr;  
      }

1. The maxwell function returns the pointer whose corresponding value is larger, so maxwell(array, &array[2]) returns array, or &array[0]. The value it points to, array[0], is set to -1. The statement ptr += 2 shifts the pointer to &array[2]. The value of the next element array[3] is changed to 9 because ptr[1] is the same as &array[2]+1, which equals &array[3]. After that, the second element array[1] is changed to 79 because &array[0]+1 equals &array[1]. So, the array so far is: {-1, 79, 4, 9, 22, 19}. Then, the program prints 3 because &array[5] - ptr is the same as &array[5] - &array[2], which equals &array[3]. The swap1 method swaps the locations of the two parameters. However, swap1(&array[0], &array[1]) passes the pointers by value, so the change only occurs within the swap1 function and nothing is changed outside. In contrast, swap2(array, &array[2]) swaps the values at &array[0] (-1) and &array[2] (4) inside the array, resulting in the final array to be: {4, 79, -1, 9, 22, 19}. Finally, each element of the array is outputted. The final output is:  
     
   3  
   4  
   79  
   -1  
   9  
   22  
   19
2. void removeS(char\* str) {  
    while (\*str != ‘\0’) {  
    if (\*str == ‘S’ || \*str == ‘s’) {  
    char\* strTemp = str;  
    while (\*str != ‘\0’) {  
    \*str = \*(str+1);  
    str++;  
    }  
    str = strTemp;  
    }  
    else  
    str++;  
    }  
   }