**Problem 1:** Assume the definitions and initializations:

char c = 'T', d = 'S';  
char \*p1 = &c;  
char \*p2 = &d;  
char \*p3;

Assume further that the address of c is 6940, the address of d is 9772, and the address of e is 2224.  
What will be printed when the following statements are executed sequentially?  
  
p3 = &d;  
cout << "\*p3 = " << \*p3 << endl; // (1)  
p3 = p1;  
cout << "\*p3 = " << \*p3  << ", p3 = " << p3 << endl; // (2)  
\*p1 = \*p2;  
cout << "\*p1 = " << \*p1 << ", p1 = " << p1 << endl; // (3)

ANSWER:

1. \*p3 = S
2. \*p3 = T, p3 = 6940
3. \*p1 = S, p1 = 6940

**Problem 2:**  Consider the following statements:  
  
int \*p;  
int i;  
int k;  
i = 42;  
k = i;  
p = &i;  
  
After these statements are executed sequentially, which of the following statements will change the value of i to 75 ?

1. k = 75;
2. \*k = 75;
3. p = 75;
4. \*p = 75;
5. Two or more of the answers will change i to 75.

ANSWER:

D.

**Problem 3:** Which of the following lines will not compile and what is the reason for that error?

char c = 'A'; // (1)

double \*p = &c; // (2)

int i = 'A'; // (3)

char \*q = &i; // (4)

char d = 65; // (5)

char \* r = &d; // (6)

int j = c; // (7)

int \* s = &j; // (8)

ANSWER:

Line (2) will not compile because the pointer p can only point to a variable of type double. Since c is a variable of type char, this line of code will not compile. Line (4) will not compile because i is not well defined(i is of type int but the code is trying to store something of type char inside). Even if i was a well-defined type int this line would not compile since the pointer q can only point to char variables. The other lines will compile since int and char are integral values.

**Problem 4:**Give the assigned value of the left-hand side variable in each assignment statement. Assume the lines are executed sequentially. Assume the address of the blocks array is 4434. To show you what I mean, the answer to the first question has been provided to you.

ANSWER:

char blocks[3] = {'A','B','C'};

char \*ptr = &blocks[0]; // ptr will be assigned the value: 4434

char temp;

temp = blocks[0]; // temp will be assigned the value: ‘A’

temp = \*(blocks + 2); // temp will be assigned the value: ‘C’

temp = \*(ptr + 1); // temp will be assigned the value: ‘B’

temp = \*ptr; // temp will be assigned the value: ‘A’

ptr = blocks + 1; // ptr will be assigned the value: 4435

temp = \*ptr; // temp will be assigned the value: ‘B’

temp = \*(ptr + 1); // temp will be assigned the value: ‘C’

ptr = blocks; // ptr will be assigned the value: 4434

temp = \*++ptr; // temp will be assigned the value: ‘B’

temp = ++\*ptr; // temp will be assigned the value: ‘C’

temp = \*ptr++; // temp will be assigned the value: ‘C’

temp = \*ptr; // temp will be assigned the value: ‘C’

**Problem 5:**  Rewrite this program and remove the variable declared at line A below.  Make your revised program generate the exact same output as the original without using the variable declared at line A.

            int num[ 6 ] = { 0, 0, 0, 0, 0, 0 };  
        int \*ptr;       // Line A  
        // rewrite all this code without this declared variable above  
        // make your revised code does the same thing as this original logic  
        ptr = num;  
        \*ptr = 100;  
        ptr ++;  
        \*ptr  = 90;  
        ptr  = &num[ 2 ];  
        \*ptr  = 80;  
        ptr  = num + 3;  
        \*ptr  = 70;  
        ptr  = num;  
        \*(ptr  + 4) = 60;  
        for (int i = 0; i <= 5; i++)  
            cout << num[ i ] << " ";  
        cout << endl;

ANSWER:

#include<cstring>

#include<iostream>

#include<string>

#include<cctype>

using namespace std;

int main()

{

int n[6] = { 0, 0, 0, 0, 0, 0 };

n[0] = 100;

n[1] = 90;

n[2] = 80;

n[3] = 70;

n[4] = 60;

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

cout << n[i] << " ";

cout << endl;

}

**Problem 6:**   Write a function named revString that accepts one character  pointer as a parameter and returns no value. The parameter must be a C-string.  This function must remove all of the digit character letters from  the string.  The revised string argument must be a valid C-string.  Your function  must not use any square brackets and must not use the strlen or strcpy library functions.  The signature for revString should be:  void revString( char \* msg )

ANSWER:

    #include<cstring>

#include<iostream>

#include<string>

#include<cctype>

using namespace std;

void revString(char\* msg);

int main()

{

char msg[100] = "Happy 2021!";

revString(msg);

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

return(0);

}

void revString(char\* msg)

{

int numberOfNotDigits = 0, posOfNull;

char swap = ' ';

bool replacedonce = false;

for (posOfNull = 0; \*(msg + posOfNull) != '\0'; posOfNull++)

{

if (!isdigit(\*(msg + posOfNull)))

{

numberOfNotDigits++;

}

}

for (int i = 0; (\*(msg + i) != '\0') && (numberOfNotDigits != 0); i++)

{

if (!isdigit(\*(msg + i)))

{

numberOfNotDigits--;

}

if (isdigit(\*(msg + i)))

{

replacedonce = false;

for (int k = i + 1; (\*(msg + k) != '\0') && (numberOfNotDigits != 0); k++)

{

if (!isdigit(\*(msg + k)) && (replacedonce == false))

{

numberOfNotDigits--;

\*(msg + i) = \*(msg + k);

for (int o = k; o < posOfNull; o++)

{

swap = \*(msg + (o + 1));

\*(msg + o) = swap;

}

posOfNull = posOfNull - 1;

break;

}

}

}

}

int print;

for (print = 0; print < posOfNull; print++)

{

if (isdigit(\*(msg + print)))

{

\*(msg + print) = '\0';

break;

}

}

}