```
#include <iostream>

using namespace std;

void deb(unsigned int v, unsigned int* bits);
unsigned int revbits(unsigned int d);
unsigned long tom_algo(int startbit, int length,
                       unsigned val, unsigned long *reg);

int main(void)
{
   static unsigned int bits[16];
   unsigned int d = 0xC5;
   deb(d, bits);
   for(int i = 15; i >=0 ; i--)
      cout << bits[i] << " ";
   cout << endl;

   cout << d << ": " << revbits(d) << endl;
   int j = 1;
   float avg[5] = {0};
   float samples[30] =
      {1,2,3,4,5,6,7,8,9,10,
       11,12,13,14,15,16,17,18,19,20,
       21,22,23,34,35,36,27,28,29,30};
   float sum = 0;

   for (int i = 0; i < 5; i++) {
      sum += samples[i];
      avg[0] = sum/5;
      j = 1;
      for(int n = 5; n < 30; n++) {
      sum = sum+samples[n] - samples[n-5];
      avg[j++] = sum/5;
      }
   }
   for (j = 0; j<5; j++)
      cout << avg[j] << ", ";

   cout << endl;

   int startbit = 9;
   int length = 8;
   unsigned long mask, temp;
   unsigned long val = 12;
   unsigned long data = 0xa0b0c345;
   unsigned long p_lsb = 0;

   // clear out all bits lower than start bit position
   mask =  0xffffffff << (startbit+1);
```

```
   cout << "mask: " << hex << mask << endl;
   p_lsb = 0xffffffff >> (32 - startbit - length); // determine the lsb
   cout << "lsb: " << hex << p_lsb << endl;
   mask += p_lsb; // clear all bits from start bit down to length
   cout << "cleared masked bits: " << hex << mask << endl;
   mask ^= 0xffffffff;  // set all masked bits
   cout << "masked bits: " << hex << mask << endl;
   temp = val;
   d = 0;
   while (temp)
   {
      d++;
      temp = temp >> 1; // find out number of bits in val
   }
   cout << "data width: " << d << endl;
   d += startbit-length;
   cout << "aligned bit-position: " << d << endl;
   if (d <= length) {
      val = val << d;
      cout << "positioned val: " << hex << val << endl;
      data += val;   // finally put positioned val into the range
      cout << "resuled data: " << hex << data << endl;
   }
   cout << endl;
   data = 0xa0b0c345;
   tom_algo(9, 4, 12, &data);
   cout << "result: " << data << endl;

return 0;
}

void deb(unsigned int v, unsigned int* bits)
{
   int i = 31;
   while (i--) {
      bits[i] = (1 & (v>>i));
   }
}

unsigned int revbits(unsigned int d)
{
   return (d ^ 0xFFFF);
}

unsigned long tom_algo(int startbit, int length,
                       unsigned val, unsigned long *reg)
{
   unsigned long mask, temp;
   unsigned long p_lsb = 0;
   int d = 0;
   unsigned long data = *reg;
   temp = val;
```

```
   while (temp)
   {
      d++;
      temp = temp >> 1; // find out number of bits in val
   }
   cout << "data width: " << d << endl;
   if (d > length)  // the data width given for the val is out of range
       return -1;

   // clear all bits from startbit position down
   mask =  0xffffffff << (startbit+1);
   cout << "mask: " << hex << mask << endl;
   p_lsb = 0xffffffff >> (32 - startbit + length-1); // determine the lsb
   cout << "lsb: " << hex << p_lsb << endl;
   temp = mask | p_lsb; // clear all bits from startbit down to length
   cout << "cleared masked bits: " << hex << temp << endl;
   cout << "original data: " << hex << data << endl;
   temp = temp & data;
   cout << "  masked copy: " << temp << endl;

   val = val << (startbit-length);
   cout << "aligned val: " << hex << val << endl;

   data = temp + val;   // finally put positioned val into the range
   *reg = data;

   return 0;   // success
}
```