```
//
//  EXAMPLE:   C++ Program to solve the 0-1 Knapsack problem
//             using backtracking.
//
#include <iostream>
using namespace std ;

// ----------------------------------------------------
class array {
   int n ;
   int * a ;
public:
   array() { n = 0 ; a = NULL ; }
   ~array(){ }
   int value(int i) ;
   int size() ;
   void readarray() ;
   void writearray() ;
} ;

int array::value(int i) { return *(a+i) ; }

void array::readarray()
{
   int i ;

   cout << "n --> " ;
   cin >> n ;
   a = new int[n] ;
   cout << "array --> " ;
   for ( i = 0 ; i < n ; i++ ) { cin >> *(a + i) ; }
}

void array::writearray()
{
   int i ;
   for ( i = 0 ; i < n ; i++ ) {
       cout << *(a + i) ;
       if ( i < (n-1) ) cout << " " ;
   }
   cout << endl ;
}

int array::size() { return n ; }

// ----------------------------------------------------
class bitvector{
   int n ;
   int * b ;
public:
   bitvector() { n = 0 ; b = NULL ; }
   ~bitvector() { }
   void init( int nn ) ;
   void set( int j ) ;
   void clear( int j ) ;
   void show( array a ) ;
} ;

void bitvector::init( int nn )
{
   int i ;
   n = nn ;
   b = new int[nn] ;
   for ( i = 0 ; i < n ; i++ )  clear(i) ;
}
```

```
void bitvector::set( int j ) { *(b+j) = 1 ; }

void bitvector::clear( int j ) { *(b+j) = 0 ; }

void bitvector::show( array a )
{
   int j ;

   cout << "{ " ;
   for ( j = 0 ; j < n ; j++ ) {
     if ( b[j] ) {
         cout << a.value(j) << " " ;
     }
   }
   cout << "}" << endl ;
}

// ------------------------------------------------------
void knap( array a, int K, bitvector & P, int m, int t )
{
   if ( t > K ) return ;
   if ( t == K ) {
     cout << "Solution found: " ;
     P.show( a ) ;
   }
   else {
      if ( m < a.size() ) {
         P.set( m ) ;
         knap( a, K, P, m+1, t + a.value(m) ) ;
         P.clear( m ) ;
         knap( a, K, P, m+1, t ) ;
      }
   }
}

// ------------------------------------------------------
void knapsack( array a, int K )
{
   bitvector P ;
   P.init( a.size() ) ;  // Make bitvector the same size as a[].
   knap( a, K, P, 0, 0 ) ;
}

// ------------------------------------------------------
main()
{
   array a ;
   int K ;

   a.readarray() ;
   cout << "K --> " ; cin >> K ;

   knapsack( a, K ) ;
}
```

```
======================== Sample Session  ========================

hoku% knap
n --> 7
array --> 3 5 7 15 19 22 30
K --> 26
Solution found: { 7 19 }
```