```cpp
//*****************************************************************************
// File name:   main.cpp
// Author:      Chadd Williams
// Date:        9/16/2013
// Class:       CS 380
// Assignment:  Vector Example
// Purpose:     Example code that stores pointers into vectors
//*****************************************************************************


#define MEM_DEBUG 1
#include "mem_debug.h"
#include <vector>
#include <iostream>
#include <ctime>

//*******************************************************
// Function:    main
//
// Description: Store pointers to ints into a vector and perform some
//              vector operations
//
// Parameters:  None
//
// Returned:    EXIT_SUCCESS on success, -1 otherwise
//
//*******************************************************
int main()
{
#ifdef MEM_DEBUG
  _CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF | _CRTDBG_LEAK_CHECK_DF);
  //_crtBreakAlloc = 147;
#endif

  /* vector of pointers */
  std::vector<int*> vecOfPointers;
  int *pInt;

  /* a forward iterator */
  std::vector<int*>::iterator iter;

  /* a reverse iterator */
  std::vector<int*>::reverse_iterator r_iter;

  srand(static_cast<unsigned int>(time(0)));

  for( int i = 0; i< 10; i++)
  {
    pInt = new int();

    // generate an integer between 0 and RAND_MAX
    *pInt = rand();

    // put the pointer on the end of the vector
    vecOfPointers.push_back(pInt);

    std::cout<< *vecOfPointers[i] << " " ;
  }


  std::cout << "\n\n VECTOR FORWARD: \n\n";

  // use the iterator to walk forward
  iter = vecOfPointers.begin();
```

```
65    while ( iter != vecOfPointers.end() )
66    {
67      // outside star: dereference the int
68      // inside star:  dereference the iterator
69      std::cout << *(*iter) << std::endl;
70
71      // increase the iterator
72      iter ++;
73    }
74
75    std::cout << "\n\n VECTOR REVERSE: \n\n";
76
77    // use the iterator to walk backwards
78    r_iter = vecOfPointers.rbegin();
79    while ( r_iter != vecOfPointers.rend() )
80    {
81      std::cout << *(*r_iter) << std::endl;
82
83      // still ++
84      r_iter ++;
85    }
86
87    std::cout << "\n\n VECTOR FORWARD: \n\n";
88    // you can also use array notation with a vector
89    for (int index = 0 ; index < vecOfPointers.size() ; index++)
90    {
91      std::cout << *(vecOfPointers[index]) << std::endl;
92    }
93
94    std::cout << "\n\n VECTOR DEALLOCATE \n\n";
95
96    for (int index = 0 ; index < vecOfPointers.size() ; index++)
97    {
98      // you can also use array notation with a vector
99      // to set values
100     delete vecOfPointers[index];
101     vecOfPointers[index] = NULL;
102   }
103
104   return EXIT_SUCCESS;
105 }
106
```