// FILE: Assign02.cpp

// An interactive test program for the IntSet data type.

#include "IntSet.h"

#include <iostream>

#include <iomanip>

#include <cstdlib>

using namespace std;

// PROTOTYPES for functions used by this test program:

void print\_menu();

// Pre: (none)

// Post: A menu of choices for this program is written to cout.

char get\_user\_command();

// Pre: (none)

// Post: The user is prompted to enter a one character command.

// The next character is read (skipping blanks and newline

// characters), and this character is returned.

int get\_object\_num(int argc);

int get\_paired\_num(int argc);

int get\_hybrid\_num(int argc);

int get\_integer(int argc);

// Pre: (none)

// Post: The user is prompted to enter an integer. The prompt

// is repeated until a valid integer can be read. The

// valid integer read is returned. The input buffer is

// cleared of any extra input until and including the

// first newline character.

void DumpDataAux(IntSet is, int objNum, ostream& out);

// Pre: (none)

// Post: Contents of is has been inserted into out following

// some custom format.

/\* Quiz: Why is is not passed by const reference? \*/

void ResetAux(IntSet& is, int objNum, ostream& out);

// Pre: (none)

// Post: is has called reset() and a message inserted into out.

int main(int argc, char\* argv[])

{

IntSet is1, is2, is3; // 3 IntSet's to perform tests on

int objectNum, // number specifying is1, is2 or is3

pairedNum, // number specifying primary and secondary objects

hybridNum, // number specifying which 1, 2 or 3 objects

givenValue; // holder for a user supplied value

char choice; // command character entered by the user

cout << "3 IntSet objects (is1 is2 is3) have been created." << endl;

do

{

if (argc == 1)

print\_menu();

choice = get\_user\_command();

switch (choice)

{

case 'a': case 'A':

objectNum = get\_object\_num(argc);

givenValue = get\_integer(argc);

switch (objectNum)

{

case 1:

cout << givenValue << (is1.add(givenValue) ? "" : " not") << " added to is1" << endl;

break;

case 2:

cout << givenValue << (is2.add(givenValue) ? "" : " not") << " added to is2" << endl;

break;

case 3:

cout << givenValue << (is3.add(givenValue) ? "" : " not") << " added to is3" << endl;

}

break;

case 'b': case 'B':

pairedNum = get\_paired\_num(argc);

switch (pairedNum)

{

case 11:

cout << "is1 is" << (is1.isSubsetOf(is1) ? "" : " not") << " subset of itself" << endl;

break;

case 12:

cout << "is1 is" << (is1.isSubsetOf(is2) ? "" : " not") << " subset of is2" << endl;

break;

case 13:

cout << "is1 is" << (is1.isSubsetOf(is3) ? "" : " not") << " subset of is3" << endl;

break;

case 21:

cout << "is2 is" << (is2.isSubsetOf(is1) ? "" : " not") << " subset of is1" << endl;

break;

case 22:

cout << "is2 is" << (is2.isSubsetOf(is2) ? "" : " not") << " subset of itself" << endl;

break;

case 23:

cout << "is2 is" << (is2.isSubsetOf(is3) ? "" : " not") << " subset of is3" << endl;

break;

case 31:

cout << "is3 is" << (is3.isSubsetOf(is1) ? "" : " not") << " subset of is1" << endl;

break;

case 32:

cout << "is3 is" << (is3.isSubsetOf(is2) ? "" : " not") << " subset of is2" << endl;

break;

case 33:

cout << "is3 is" << (is3.isSubsetOf(is3) ? "" : " not") << " subset of itself" << endl;

}

break;

case 'c': case 'C':

objectNum = get\_object\_num(argc);

givenValue = get\_integer(argc);

switch (objectNum)

{

case 1:

cout << givenValue << " is" << (is1.contains(givenValue) ? "" : " not") << " in is1" << endl;

break;

case 2:

cout << givenValue << " is" << (is2.contains(givenValue) ? "" : " not") << " in is2" << endl;

break;

case 3:

cout << givenValue << " is" << (is3.contains(givenValue) ? "" : " not") << " in is3" << endl;

}

break;

case 'd': case 'D':

hybridNum = get\_hybrid\_num(argc);

/\* Quiz: Why is the following block written in such

a weird-looking fashion? \*/

{

{

IntSet tccSet1 = is1;

IntSet tccSet2 = is2;

IntSet tccSet3 = is3;

tccSet1.reset();

tccSet2.reset();

tccSet3.reset();

}

{

IntSet taoSet1;

IntSet taoSet2;

IntSet taoSet3;

taoSet1 = is1;

taoSet2 = is2;

taoSet3 = is3;

}

switch (hybridNum)

{

case 1:

DumpDataAux(is1, 1, cout);

break;

case 2:

DumpDataAux(is2, 2, cout);

break;

case 3:

DumpDataAux(is3, 3, cout);

break;

case 12:

DumpDataAux(is1, 1, cout);

DumpDataAux(is2, 2, cout);

break;

case 13:

DumpDataAux(is1, 1, cout);

DumpDataAux(is3, 3, cout);

break;

case 23:

DumpDataAux(is2, 2, cout);

DumpDataAux(is3, 3, cout);

break;

case 123:

DumpDataAux(is1, 1, cout);

DumpDataAux(is2, 2, cout);

DumpDataAux(is3, 3, cout);

}

}

break;

case 'e': case 'E':

pairedNum = get\_paired\_num(argc);

switch (pairedNum)

{

case 11:

cout << ( (is1 == is1) ? "is1 is equal to itself"

: "is1 is not equal to itself" ) << endl;

break;

case 12:

cout << ( (is1 == is2) ? "is1 is equal to is2"

: "is1 is not equal to is2" ) << endl;

break;

case 13:

cout << ( (is1 == is3) ? "is1 is equal to is3"

: "is1 is not equal to is3" ) << endl;

break;

case 21:

cout << ( (is2 == is1) ? "is2 is equal to is1"

: "is2 is not equal to is1" ) << endl;

break;

case 22:

cout << ( (is2 == is2) ? "is2 is equal to itself"

: "is2 is not equal to itself" ) << endl;

break;

case 23:

cout << ( (is2 == is3) ? "is2 is equal to is3"

: "is2 is not equal to is3" ) << endl;

break;

case 31:

cout << ( (is3 == is1) ? "is3 is equal to is1"

: "is3 is not equal to is1" ) << endl;

break;

case 32:

cout << ( (is3 == is2) ? "is3 is equal to is2"

: "is3 is not equal to is2" ) << endl;

break;

case 33:

cout << ( (is3 == is3) ? "is3 is equal to itself"

: "is3 is not equal to itself" ) << endl;

}

break;

case 'i': case 'I':

pairedNum = get\_paired\_num(argc);

switch (pairedNum)

{

case 11:

is1 = is1.intersect(is1);

cout << "is1 has been intersected with itself" << endl;

break;

case 12:

is1 = is1.intersect(is2);

cout << "is1 has been intersected with is2" << endl;

break;

case 13:

is1 = is1.intersect(is3);

cout << "is1 has been intersected with is3" << endl;

break;

case 21:

is2 = is2.intersect(is1);

cout << "is2 has been intersected with is1" << endl;

break;

case 22:

is2 = is2.intersect(is2);

cout << "is2 has been intersected with itself" << endl;

break;

case 23:

is2 = is2.intersect(is3);

cout << "is2 has been intersected with is3" << endl;

break;

case 31:

is3 = is3.intersect(is1);

cout << "is3 has been intersected with is1" << endl;

break;

case 32:

is3 = is3.intersect(is2);

cout << "is3 has been intersected with is2" << endl;

break;

case 33:

is3 = is3.intersect(is3);

cout << "is3 has been intersected with itself" << endl;

}

break;

case 'k': case 'K':

objectNum = get\_object\_num(argc);

givenValue = get\_integer(argc);

switch (objectNum)

{

case 1:

cout << givenValue << (is1.remove(givenValue) ? " removed from" : " not found in") << " is1" << endl;

break;

case 2:

cout << givenValue << (is2.remove(givenValue) ? " removed from" : " not found in") << " is2" << endl;

break;

case 3:

cout << givenValue << (is3.remove(givenValue) ? " removed from" : " not found in") << " is3" << endl;

}

break;

case 'm': case 'M':

hybridNum = get\_hybrid\_num(argc);

switch (hybridNum)

{

case 1:

cout << " is1 is" << (is1.isEmpty() ? "" : " not") << " empty" << endl;

break;

case 2:

cout << " is2 is" << (is2.isEmpty() ? "" : " not") << " empty" << endl;

break;

case 3:

cout << " is3 is" << (is3.isEmpty() ? "" : " not") << " empty" << endl;

break;

case 12:

cout << " is1 is" << (is1.isEmpty() ? "" : " not") << " empty" << endl;

cout << " is2 is" << (is2.isEmpty() ? "" : " not") << " empty" << endl;

break;

case 13:

cout << " is1 is" << (is1.isEmpty() ? "" : " not") << " empty" << endl;

cout << " is3 is" << (is3.isEmpty() ? "" : " not") << " empty" << endl;

break;

case 23:

cout << " is2 is" << (is2.isEmpty() ? "" : " not") << " empty" << endl;

cout << " is3 is" << (is3.isEmpty() ? "" : " not") << " empty" << endl;

break;

case 123:

cout << " is1 is" << (is1.isEmpty() ? "" : " not") << " empty" << endl;

cout << " is2 is" << (is2.isEmpty() ? "" : " not") << " empty" << endl;

cout << " is3 is" << (is3.isEmpty() ? "" : " not") << " empty" << endl;

}

break;

case 'r': case 'R':

hybridNum = get\_hybrid\_num(argc);

switch (hybridNum)

{

case 1:

ResetAux(is1, 1, cout);

break;

case 2:

ResetAux(is2, 2, cout);

break;

case 3:

ResetAux(is3, 3, cout);

break;

case 12:

ResetAux(is1, 1, cout);

ResetAux(is2, 2, cout);

break;

case 13:

ResetAux(is1, 1, cout);

ResetAux(is3, 3, cout);

break;

case 23:

ResetAux(is2, 2, cout);

ResetAux(is3, 3, cout);

break;

case 123:

ResetAux(is1, 1, cout);

ResetAux(is2, 2, cout);

ResetAux(is3, 3, cout);

}

break;

case 's': case 'S':

pairedNum = get\_paired\_num(argc);

switch (pairedNum)

{

case 11:

is1 = is1.subtract(is1);

cout << "is1 has been subtracted from itself" << endl;

break;

case 12:

is1 = is1.subtract(is2);

cout << "is2 has been subtracted from is1" << endl;

break;

case 13:

is1 = is1.subtract(is3);

cout << "is3 has been subtracted from is1" << endl;

break;

case 21:

is2 = is2.subtract(is1);

cout << "is1 has been subtracted from is2" << endl;

break;

case 22:

is2 = is2.subtract(is2);

cout << "is2 has been subtracted from itself" << endl;

break;

case 23:

is2 = is2.subtract(is3);

cout << "is3 has been subtracted from is2" << endl;

break;

case 31:

is3 = is3.subtract(is1);

cout << "is1 has been subtracted from is3" << endl;

break;

case 32:

is3 = is3.subtract(is2);

cout << "is2 has been subtracted from is3" << endl;

break;

case 33:

is3 = is3.subtract(is3);

cout << "is3 has been subtracted from itself" << endl;

}

break;

case 'u': case 'U':

pairedNum = get\_paired\_num(argc);

switch (pairedNum)

{

case 11:

is1 = is1.unionWith(is1);

cout << "is1 has been unioned with itself" << endl;

break;

case 12:

is1 = is1.unionWith(is2);

cout << "is1 has been unioned with is2" << endl;

break;

case 13:

is1 = is1.unionWith(is3);

cout << "is1 has been unioned with is3" << endl;

break;

case 21:

is2 = is2.unionWith(is1);

cout << "is2 has been unioned with is1" << endl;

break;

case 22:

is2 = is2.unionWith(is2);

cout << "is2 has been unioned with itself" << endl;

break;

case 23:

is2 = is2.unionWith(is3);

cout << "is2 has been unioned with is3" << endl;

break;

case 31:

is3 = is3.unionWith(is1);

cout << "is3 has been unioned with is1" << endl;

break;

case 32:

is3 = is3.unionWith(is2);

cout << "is3 has been unioned with is2" << endl;

break;

case 33:

is3 = is3.unionWith(is3);

cout << "is3 has been unioned with itself" << endl;

}

break;

case 'z': case 'Z':

hybridNum = get\_hybrid\_num(argc);

switch (hybridNum)

{

case 1:

cout << " is1 has " << is1.size() << " items" << endl;

break;

case 2:

cout << " is2 has " << is2.size() << " items" << endl;

break;

case 3:

cout << " is3 has " << is3.size() << " items" << endl;

break;

case 12:

cout << " is1 has " << is1.size() << " items" << endl;

cout << " is2 has " << is2.size() << " items" << endl;

break;

case 13:

cout << " is1 has " << is1.size() << " items" << endl;

cout << " is3 has " << is3.size() << " items" << endl;

break;

case 23:

cout << " is2 has " << is2.size() << " items" << endl;

cout << " is3 has " << is3.size() << " items" << endl;

break;

case 123:

cout << " is1 has " << is1.size() << " items" << endl;

cout << " is2 has " << is2.size() << " items" << endl;

cout << " is3 has " << is3.size() << " items" << endl;

}

break;

case 'q': case 'Q':

cout << "Quit option selected...bye" << endl;

break;

default:

cout << choice << " is not a valid option...try again"

<< endl;

}

}

while (choice != 'q' && choice != 'Q');

cin.ignore(999, '\n');

cout << "Press Enter or Return when ready...";

cin.get();

return EXIT\_SUCCESS;

}

void print\_menu()

{

cout << endl;

cout << "The following choices are available: " << endl;

cout << " a Add an item to is1, is2 or is3" << endl;

cout << " b Query if 1 of is1, is2 or is3 is subset of is1, is2 or is3" << endl;

cout << " c Query if an item is in is1, is2 or is3" << endl;

cout << " d Display 1 or more of is1, is2 and is3 (to stdout)" << endl;

cout << " e Query if 1 of is1, is2 or is3 is equal to is1, is2 or is3" << endl;

cout << " i Intersect 1 of is1, is2 or is3 with is1, is2 or is3" << endl;

cout << " k Remove an item from is1, is2 or is3" << endl;

cout << " m Query if 1 or more of is1, is2 and is3 is/are empty" << endl;

cout << " r Reset (make empty) 1 or more of is1, is2 and is3" << endl;

cout << " s Subtract 1 of is1, is2 or is3 from is1, is2 or is3" << endl;

cout << " u Union 1 of is1, is2 or is3 with is1, is2 or is3" << endl;

cout << " z Query # of items in 1 or more of is1, is2 and is3" << endl;

cout << " q Quit this test program" << endl;

}

char get\_user\_command()

{

char command;

cout << "Enter choice: ";

cin >> command;

cout << command << " read." << endl;

return command;

}

int get\_object\_num(int argc)

{

int result;

cout << "Enter object # (1 = is1, 2 = is2, 3 = is3) ";

cin >> result;

while ( ! cin.good() )

{

cerr << "Bad integer input..." << endl;

cin.clear();

cin.ignore(999, '\n');

cout << "Re-enter object # (1 = is1, 2 = is2, 3 = is3) ";

cin >> result;

}

if (argc < 2)

cin.ignore(999, '\n');

while (result != 1 && result != 2 && result != 3)

{

cerr << "Bad object # (must be 1, 2 or 3)..." << endl;

cout << "Re-enter object # (1 = is1, 2 = is2, 3 = is3) ";

cin >> result;

while ( ! cin.good() )

{

cerr << "Bad integer input..." << endl;

cin.clear();

cin.ignore(999, '\n');

cout << "Re-enter object # (1 = is1, 2 = is2, 3 = is3) ";

cin >> result;

}

cin.ignore(999, '\n');

}

cout << result << " read." << endl;

return result;

}

int get\_paired\_num(int argc)

{

int result;

cout << "Enter object\_pair # (12 for is1.OP(is2), 32 for is3.OP(is2),...) ";

cin >> result;

while ( ! cin.good() )

{

cerr << "Bad integer input..." << endl;

cin.clear();

cin.ignore(999, '\n');

cout << "Re-enter object\_pair # (12 for is1.OP(is2), 32 for is3.OP(is2),...) ";

cin >> result;

}

if (argc < 2)

cin.ignore(999, '\n');

while (result != 11 && result != 12 && result != 13 &&

result != 21 && result != 22 && result != 23 &&

result != 31 && result != 32 && result != 33)

{

cerr << "Bad object\_pair # (must be 11, 12, 13, 21, 22, 23, 31, 32 or 33)..." << endl;

cout << "Re-enter object\_pair # (12 for is1.OP(is2), 32 for is3.OP(is2),...) ";

cin >> result;

while ( ! cin.good() )

{

cerr << "Bad integer input..." << endl;

cin.clear();

cin.ignore(999, '\n');

cout << "Re-enter object\_pair # (12 for is1.OP(is2), 32 for is3.OP(is2),...) ";

cin >> result;

}

cin.ignore(999, '\n');

}

cout << result << " read." << endl;

return result;

}

int get\_hybrid\_num(int argc)

{

int result;

cout << "Enter hybrid # (1 for is1, 23 for is2 and is3, 123 for is1, is2 and is3,...) ";

cin >> result;

while ( ! cin.good() )

{

cerr << "Bad integer input..." << endl;

cin.clear();

cin.ignore(999, '\n');

cout << "Re-enter hybrid # (1 for is1, 23 for is2 and is3, 123 for is1, is2 and is3,...) ";

cin >> result;

}

if (argc < 2)

cin.ignore(999, '\n');

while (result != 1 && result != 2 && result != 3 &&

result != 12 && result != 13 && result != 23 &&

result != 123)

{

cerr << "Bad object\_pair # (must be 1, 2, 3, 12, 13, 23 or 123)..." << endl;

cout << "Re-enter hybrid # (1 for is1, 23 for is2 and is3, 123 for is1, is2 and is3,...) ";

cin >> result;

while ( ! cin.good() )

{

cerr << "Bad integer input..." << endl;

cin.clear();

cin.ignore(999, '\n');

cout << "Re-enter hybrid # (1 for is1, 23 for is2 and is3, 123 for is1, is2 and is3,...) ";

cin >> result;

}

cin.ignore(999, '\n');

}

cout << result << " read." << endl;

return result;

}

int get\_integer(int argc)

{

int result;

cout << "Enter integer value ";

cin >> result;

while ( ! cin.good() )

{

cerr << "Bad integer input..." << endl;

cin.clear();

cin.ignore(999, '\n');

cout << "Re-enter integer value ";

cin >> result;

}

if (argc < 2)

cin.ignore(999, '\n');

cout << result << " read." << endl;

return result;

}

void DumpDataAux(IntSet is, int objNum, ostream& out)

{

if ( is.isEmpty() )

out << " is" << objNum << ": (empty)" << endl;

else

{

out << " is" << objNum << ": ";

is.DumpData(out);

out << endl;

}

}

void ResetAux(IntSet& is, int objNum, ostream& out)

{

is.reset();

out << " is" << objNum << " has been reset and is now empty" << endl;

}