UNIVERSIDAD DE COSTA RICA
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA
**ESTRUCTURAS ABSTRACTAS DE DATOS**
**Y ALGORITMOS PARA INGENIERÍA**
IE-0217 - II CICLO 2013
LABORATORIO 7

1. Compare exhaustivamente las estructuras de datos y algoritmos (DSA) de la librería estándar de C++ con las DSA de la anterior Standard Template Library (STL). Tabule sus resultados.

2. Uso de algoritmos básicos:

   *a*) Utilice el algoritmo *copy* para copiar los enteros aleatorios de un array de 50 elementos (en un rango $[10, 20]$) a un vector.

   *b*) Utilice *sort* para ordenar ambas secuencias de manera no decreciente.

   *c*) Utilice de nuevo *copy* para imprimir el contenido en pantalla luego de ordenar ambas secuencias.

   *d*) Utilice *greater* como parámetro de *sort* para ordenar las secuencias de manera no creciente.

   *e*) Utilice *adjacent_find* para encontrar e imprimir cada rango de elementos repetidos en una línea separada.

   *f*) Use *count* y *accumulate* para encontrar la mediana, media, varianza y moda de los enteros contenidos.

3. ¿Para qué sirve el algoritmo *reverse()*? Escriba su propia version usando el siguiente esqueleto:

```
template <typename BidirectionalIterator>
void my_reverse(BidirectionalIterator first, BidirectionalIterator last) {
    // implementacion aqui
    //La ausencia de tildes se debe a una limitante del paquete
    //lstlisting de latex.
    //Puntos extras por proveer una solucion
}
```

   *a*) Escriba una función test(int size), que cree un vector de enteros aleatorios, los imprima en pantalla, llame a su función *my_reverse* y los imprima de nuevo en pantalla. La función debe poder ejecutarse de la siguiente forma: *my_reverse(v.begin(), v.end());*

   *b*) Debe probar con vectores con una cantidad de elementos par e impar. Con un vector de longitud 0 y 1.

4. Existen básicamente 5 tipos de iteradores: Input, Output, Forward, Bidirectional, Random Access.

   *a*) Descríbalos en detalle.

   *b*) ¿Cuáles contenedores poseen cuáles iteradores y porqué? Cree una tabla con los resultados obtenidos.

   *c*) Escriba un programa sofisticado, que resuelva un problema creativo, en el cual se presente el uso apropiado de cada uno de los iteradores anteriores.

5. Observe y analice el siguiente código:

```
vector<int> v;
v.push_back(1);
v.push_back(4);
v.push_back(2);
```

```
v.push_back(8);
v.push_back(5);
v.push_back(7);

copy(v.begin(), v.end(), ostream_iterator<int>(cout, " "));
cout << endl;

vector<int>::iterator new_end =
        remove_if(v.begin(), v.end(),
                  compose1(bind2nd(equal_to<int>(), 0),
                           bind2nd(modulus<int>(), 2)));
copy(v.begin(), v.end(), ostream_iterator<int>(cout, " "));
cout << endl;
```

*a*) ¿Qué busca el autor con este código?

*b*) ¿Qué hace el código?

*c*) ¿Porqué este código no está bien?

*d*) ¿Cómo arreglarlo? Arréglelo

6. Cree un programa que simule el juego de pocker. Utilice para su solución los algoritmos y estructuras de la librería estándar. (El ambiente gráfico NO es necesario).