CC3001 Algoritmos y Estructuras de Datos
Pauta Auxiliar 1

Prof. Jérémy Barbay

Aux. Manuel Olguín

Problema 1

Solución Algoritmo 1:

- Invariante: Supongamos largo de string n. Usar dos variables i, j. i indica posición en el string partiendo de la izquierda, j indica posición en el string partiendo de la derecha hacia atrás. El invariante es: "$i < j$, substring $S[0..i-1]$ es el inverso de substring $S[j+1..n-1]$".
- Condiciones iniciales: $i = 0$, $j = n-1$ (el substring izquierdo es vacío y por lo tanto es el inverso del substring derecho que también es vacío).
- Condiciones de término:
  - $i = j$ (hay un sólo carácter en la mitad del string y el substring anterior es inverso al substring posterior, por lo tanto el string es palíndrome)
  - $i > j$ (ya se compararon todos los caracteres hasta la mitad, por lo tanto el string es palíndrome).

- Cuerpo del ciclo:

```
i = 0;
j = n-1;
test = 1;
while(i < j) {
  if(s.charAt(i) != s.charAt(j)){
    test = 0;
    break;
  }
  i++;
  j++;
}

if(test) {
  System.out.println("Es palindrome");
}
else {
  System.out.println("No es palindrome");
}
```

Solución Algoritmo 2:

- Invariante: Supongamos largo de String n. Usar dos variables i, j. i indica posición en el String partiendo de la izquierda, j indica posición en el String partiendo de la derecha hacia atrás. El invariante es "$i \leq j$, número de consonantes hasta posición $i-1$ es el mismo que desde $j+1$ hasta el final del String, y todas las consonantes hasta la posición $i-1$ del String se intercambiaron con las consonantes desde $j+1$".
- Condiciones iniciales: $i = 0$, $j = n-1$ (no hay consonantes antes de i ni después de j, por lo que el invariante es válido).
- Condición de término: $i = j$ (si quedara una consonante en dicho casillero no importa, se queda ahí mismo).

- Cuerpo del ciclo:

```
i = 0;
j = n-1;
while (i < j) {
  while (i < j && s.charAt(i) no es consonante){
    i++; // puede romper invariante
  }
  if (i == j) {
    break;
  }
  while (i < j && s.charAt(j) no es consonante) {
    j--; // puede romper invariante
  }
  if (i == j) {
    break;
  }
  intercambiar(i,j); // Recupera invariante
  i++;
  j--;
}
```

Problema 2

Solución:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Ackermann {

    public static int ackermann ( int m, int n )
    {
        if ( m == 0 )
            return n + 1;
        else if ( n == 0 )
            return ackermann( m - 1, 1);
        else
            return ackermann( m - 1, ackermann( m, n - 1));
    }

    public static void main ( String[] args ) throws IOException {

        BufferedReader in = new BufferedReader ( new InputStreamReader( System.in ) )
        String input;
        String output = "";
        String[] mn = new String[2]; //Entrada separada

        while ( ( input = in.readLine () ) != null )
        //leemos linea por linea
        //para terminar de entregar entrada, hay que
        //enviar "fin de documento" -> Ctrl + D
        {
            mn = input.split(" ");
            int M = Integer.parseInt(mn[0]);
            int N = Integer.parseInt(mn[1]);
            output += ackermann(M, N) + "\n";
        }
```

```
        System.out.print ( output );
        //luego, lo imprimimos de una
        System.out.flush ();
        in.close ();

    }

}
```