UNIVERSIDAD NACIONAL
Paradigmas de Programación
Proyecto de Investigación
Prof. Eddy Ramirez
David Lobo Cristian Díaz

Problema 3714- Royale with Cheese

1 Especificación del problema

El problema trata de realizar una codificación de una cadena de carácteres. En donde se analiza uno por uno los caracteres, creando la codificación númerica de cada uno de ellos. Si este ya fue revisado, simplemente se sustituye por el número ya asignado. Al final el programa imprime la codificación final de la cadena.

2 Especificación de entrada:

La entrada consiste en múltiples casos de prueba. Cada caso de prueba contiene una cadena S con longitud n ($1 \leq n \leq 10\wedge5$) a que identifica una casa en el extraño sistema de numeración implementado en Amsterdam. La cadena S se forma solo con 26 letras minúsculas (de a a z).

3 Especificación de salida:

Para cada identificador de una casa en el sistema de numeración de Amsterdam, tiene que generar una versión codificada de la ID, tal como lo hace habitualmente la policía. Cada salida va en una línea.

4 Condiciones del problema:

El problema contiene una condición para la codificación final. Esta es, se tiene que sustituir el **2** por un **5**, **6** por un **9**; así también en su revéz el **5** por un **2** y un **9** por el **6**.

5 Ejemplo de Entrada y Salida:

Entrada	Salida
abbchocx	15534239
maplortycbjce	1534297861011615

6 Resolución del problema:

El primer paso es recibir la cadena de carácteres. Y se analiza dos vectores auxiliares Vector Letras**(VL)** y Vector Salida **(VS)**.

Al recibir la cadena, se empieza analizar cada uno de los caracteres, se inicializa el VL en donde se va añadiendo si el caracter se encuentra en él. Si el caracter ya existe, simplemente se

agrega en el número de posición en el VS. Hay que tomar en cuenta que durante la codificación se va realizando el parseo de las posiciones.

Dando como resultado final la cadena codificada.

6.1 Método de Parseo:

1. Si el número es mayor o igual a 20, retorna el **número + 30** .
2. Si el número ingresado módulo de 10 es igual 2, retorna el **número + 3** .
3. Si el número ingresado módulo de 10 es igual 6, retorna el **número + 3** .
4. Si el número ingresado módulo de 10 es igual 5, retorna el **número - 3** .
5. Si el número ingresado módulo de 10 es igual 9, retorna el **número - 3** .
6. Si no cumple con ninguno de los casos, retorna el **número** .

7 Aprobación de Algoritmo:

Aprobación de la página **Caribbean Online Judge** `http://coj.uci.cu`

lobo24	3714	**Accepted**	426	1 MB	2 KB	C

8 Soluciónes Programadas

1. ***Lenguaje Programación C***.

   Método Principal

```c
int main(int argc, char** argv) {
    var = (char*) malloc(sizeof (char));

    while (scanf("%s", var) != EOF) {
      fflush(stdin);
      fflush(stdout);

      cnt = 0;
      for (int i = 0; i < TAM; i++) {
        vecAux[i] = 0;
      }

      j = 0;
      for (int i = 0; i < strlen(var); i++) {
        if (!vecAux[var[i] - 'a']) {
          vecAux[var[i] - 'a'] = ++cnt;
        }

        if (vecAux[var[i] - 'a'] < 10) {
          vecOut[j++] = vecAux[var[i] - 'a'] + '0';
        } else {
          vecOut[j++] = vecAux[var[i] - 'a'] / 10 + '0';
          vecOut[j++] = vecAux[var[i] - 'a'] % 10 + '0';
        }
      }

      vecOut[j] = '\0';
      parseo();
      printf("%d \n", vecOut);
      free(var);
    }

  free(var);
  return 0;
}
```

   Método Parseo

```
int parseo() {
    for (int i = 0; i < j; i++) {
        if (vecOut[i] == '5') {
            vecOut[i] = '2';
        } else if (vecOut[i] == '2') {
            vecOut[i] = '5';
        } else if (vecOut[i] == '6') {
            vecOut[i] = '9';
        } else if (vecOut[i] == '9') {
            vecOut[i] = '6';
        }
    }
}
```

2. ***Lenguaje Programación Scheme***.

Método Principal

```
(define royale
  (lambda(L)
  (cond ((null? L) '())
        ((= 1 (length L)) (list 1))
        (else (royale-aux (cdr L) (list (car L)) (list 1) 1)))))

(define royale-aux
  (lambda (RL LV LS Count)
    (cond ((null? RL) LS)
          ((es-miembro? (car RL) LV)
            (royale-aux (cdr RL) LV
              (append LS
                (list (parseo(get-pos (car RL) LV 1)))) Count))
          (else (royale-aux (cdr RL)
            (append LV (list (car RL)))
            (append LS (list (parseo (+ Count 1)))) (+ Count 1))))))
```

Método Parseo

```
(define parseo
  (lambda(pos-elem)
    (cond ((>= pos-elem 20) (+ pos-elem 30))
          ((= (remainder pos-elem 10) 2) (+ pos-elem 3))
          ((= (remainder pos-elem 10) 6) (+ pos-elem 3))
          ((= (remainder pos-elem 10) 5) (- pos-elem 3))
          ((= (remainder pos-elem 10) 9) (- pos-elem 3))
          (else pos-elem))))
```

Método devuelve elemento y es miembro

```
(define es-miembro?
  (lambda (elem l)
    (ormap (lambda (x) (equal? elem x)) l)))

(define get-pos
   (lambda (elem l cont)
     (cond ((null? l) '())
           ((eq? (car l) elem) cont)
           (else (get-pos elem (cdr l) (+ cont 1))))))
```

3. ***Lenguaje Programación Erlang***.

Método Principal

```
% principal
royale([])->[];
royale([H|T])->royale(T, [H], [1], 1).

% (RL LV LS Count)
royale([],LV,LS,C)->lists:concat(LS);
royale([H|T],LV,LS,C)->royale([H|T],LV,LS,C,esMiembro(H,LV)).

%Cond
royale([H|T],LV,LS,C,true)->royale(T, LV, LS++[parseo(getPos(H, LV, 1))], C);
royale([H|T],LV,LS,C,false)->royale(T, LV++[H], LS++[parseo(C+1)], C+1).
```

Método Parseo

```
% parseo
parseo(X)when (X >= 20)->X+30;
parseo(X)when (X rem 10 == 2)-> X+3;
parseo(X)when (X rem 10 == 6)-> X+3;
parseo(X)when (X rem 10 == 5)-> X-3;
parseo(X)when (X rem 10 == 9)-> X-3;
parseo(X)->X.
```

Método devuelve elemento y es miembro

```
%esMiembro?
esMiembro(X,[])->false;
esMiembro(X,[X|_T])->true;
esMiembro(X,[H|T])->esMiembro(X, T).

% get-pos
getPos(X,[],C)->[];
getPos(X,[X|_T],C)->C;
getPos(X,[H|T],C)->getPos(X,T,C+1).
```

4. ***Lenguaje Programación Java***.

Método Principal

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

    royalewithcheese m = new royalewithcheese();
    Scanner sc = new Scanner(System.in);

    while (sc.hasNext()) {
        System.out.println(m.coding(sc.nextLine()));
    }
}
```

Método Parseo

```java
public int parseo(int n) {
    if (n >= 20) {
        n += 30;
    }
    if (n % 10 == 2 || n % 10 == 6) {
        return n + 3;
    }
    if (n % 10 == 5 || n % 10 == 9) {
        return n - 3;
    }
    return n;
}
```

Método HashTable

```java
public String coding(String input) {
    Map<Character, String> map = new HashMap<Character, String>();
    String output = "";
    int size = input.length();
    if (size < 100000 && input.matches("[a-z]+")) {
        int i = 0, j = 1;
        for (; i < size; i++) {
            if (!map.containsKey(input.charAt(i))) {
                map.put(input.charAt(i), parseo(j++) + "");
                output += map.get(input.charAt(i));
            } else {
                output += map.get(input.charAt(i));
            }
        }
    }
    return output;
}
```

5. ***Lenguaje Programación Prolog***.

Método Principal

```
royale([],[]):-!.
royale([A,B|C],R):- esMiembro(B,[A],X),royale([B|C],C,[A],1,X,[1],R).
royale([A],[],_,Pos,0,Lf,R):- Pos1 is Pos+1, parseo(Pos1,X),
                              append(Lf,[X],R),!.
royale([A],[],Nw,_,1,Lf,R):- getPos(A,Nw,P), parseo(P,X),
                             append(Lf,[X],R),!.

royale([A|B],[C|D],Nw,Pos,1,Lf,R):- getPos(A,Nw,P), parseo(P,X),
                    append(Lf,[X],Lf1),esMiembro(C,Nw,M),
                    royale(B,D,Nw,Pos,M,Lf1,R).

royale([A|B],[C|D],Nw,Pos,0,Lf,R):- append(Nw,[A],Nw1),Pos1 is Pos+1,
                    parseo(Pos1,X),append(Lf,[X],Lf1),esMiembro(C,Nw1,M),
                    royale(B,D,Nw1,Pos1,M,Lf1,R).
```

Método Parseo

```
parseo(X, Y):- X>=20, X=<27, X1 is X+30, parseo(X1,Y), !.
parseo(X, Y):- 2 is mod(X, 10), Y is X+3, !.
parseo(X, Y):- 6 is mod(X, 10), Y is X+3, !.
parseo(X, Y):- 5 is mod(X, 10), Y is X-3, !.
parseo(X, Y):- 9 is mod(X, 10), Y is X-3, !.
parseo(X,Y):- Y is X, !.
```

Método devuelve elemento y es miembro

```
% esMiembro?
esMiembro(A,[],0):-!.
esMiembro(A,[A|_T],1):-!.
esMiembro(A,[_H|T],R):-esMiembro(A,T,R).

% retornar posicion
getPos(E,[E|_],1):-!.
getPos(E,[E|_],P):-P,!.
getPos(E,[_|C],P):- getPos(E,C,P1), P is P1 + 1.
```