Tipos Abstractos de Datos Segunda relación de problemas Tema 1 (2001-02)

1.– Verificar formalmente los siguientes programas:

a) $\{Q \equiv n \geq 0\}$

```
i:=1;
WHILE (i<=n) AND (a[i]#x) DO i:=i+1 END;
encontrado:=(i<=n);
```

$\{R \equiv encontrado = (\exists \alpha \in \{1..n\} \bullet ((a[\alpha] = x) \wedge (\forall \beta \in \{1..\alpha - 1\} \bullet a[\beta] \neq x)))\}$

b) $\{Q \equiv (x = A) \wedge (y = B)\}$

```
t:=x;
x:=y;
y:=t;
```

$\{R \equiv (x = B) \wedge (y = A)\}$

c) $\{Q \equiv (a > 0) \wedge (b > 0)\}$

```
x:=a;y:=b;
WHILE (x#y) DO
  IF x<y THEN y:=y-x
  ELSE x:=x-y END;
END;
```

$\{R \equiv x = mcd(a, b)\}$

d) $\{Q \equiv x \geq 0\}$

```
u:=0;v:=1;w:=1;
WHILE w<=x DO
  u:=u+1; v:=v+2; w:=w+v
END;
```

$\{R \equiv ((u * u) \leq x) \wedge (x < (u + 1) * (u + 1)\}$

e) $\{Q \equiv n \geq 0\}$

```
i:=1; encontrado:=FALSE;
WHILE (i<=n) AND (NOT encontrado) DO
  IF a[i]=x THEN encontrado:=TRUE END
END;
```

$\{R \equiv encontrado = (\exists \alpha \in \{1..n\} \bullet ((a[\alpha] = x) \wedge (\forall \beta \in \{1..\alpha - 1\} \bullet a[\beta] \neq x)))\}$

f) $\{Q \equiv (a \geq 0) \wedge (b \geq 0)\}$

```
x:=a;y:=b;z:=1;
WHILE y>0 DO
  IF (y MOD 2)=0 THEN y:=y DIV 2; x:=x*x
  ELSE y:=y-1; z:=z*x END
END;
```

$\{R \equiv (z = a^b)\}$

g) $\{Q \equiv n \geq 1\}$

```
i:=2; min:=a[1];
WHILE (i<=n) DO
  IF a[i]<min THEN min:=a[i] END;
  i:=i+1
END;
```

$\{R \equiv (\forall \alpha \in \{1..n\} \bullet (a[\alpha] \geq min)) \wedge (\exists \beta \in \{1..n\} \bullet a[\beta] = min))\}$

h) $\{Q \equiv n \geq 1\}$

```
sum:=a[1];
FOR k:=2 TO n DO
  sum:=sum+a[k]
END;
```

$\{R \equiv sum = (\Sigma \alpha \in \{1..n\} \bullet (a[\alpha]))\}$

i) $\{Q \equiv n \geq 1\}$

```
i:=1; j:=n; capicua:=TRUE;
WHILE (i<j) AND capicua DO
  capicua:=(a[i]=a[j]);
  i:=i+1; j:=j-1
END;
```

$\{R \equiv capicua = (\forall \alpha \in \{1..nDIV2\} \bullet (a[\alpha] = a[n - \alpha + 1]))\}$

j) $\{Q \equiv (n \geq 0) \wedge (\forall \alpha, \beta \in \{1..n\} \bullet (\alpha < \beta \Rightarrow a[\alpha] \leq a[\beta]))\}$

```
PROCEDURE seg(a:vector; n:CARDINAL) (* p *):CARDINAL;
  VAR i,p:CARDINAL;
BEGIN
  i:=1; p:=1;
  WHILE (i<=n) DO
    IF a[i-p]=a[i] THEN p:=p+1 END;
    i:=i+1
  END;
  RETURN p
END seg;
```

$\{R \equiv (\exists \alpha \in \{1..n-p+1\} \bullet (a[\alpha] = a[\alpha+p-1])) \wedge (\forall \alpha \in \{1..n-p\} \bullet (a[\alpha] \neq a[\alpha+p]))\}$

2.– Dado un vector $a[1..n]$ de enteros con $n \geq 1$, diseñar una función que calcule su máximo y verifíquese formalmente.

3.– Diseñar y verificar una función que calcule la moda de un vector $a[1..n]$ de enteros, con $n \geq 1$.

4.– *El rellano más largo (E.Gries).* Dado un vector $a[1..n]$ no decreciente de enteros con $n \geq 1$, diseñar una función que calcule la longitud de su rellano más largo. Un rellano es un tramo de valores consecutivos iguales. Verificar formalmente dicha función.

5.– *La bandera holandesa (E.W.Dijkstra).* Se dispone de una hilera de n bolas azules, rojas y blancas, con $n \geq 0$. Se dispone también de un brazo mecánico para manipular las bolas que puede realizar, bajo control de un programa, las dos funciones siguientes:

a) `color(i)`, con $1 \leq i \leq n$, que devuelve el color de la bola i.

b) `permuta(i,j)`, con $1 \leq i,j \leq n$, que intercambia de posición las bolas i y j.

Diseñar y verificar un algoritmo que coloque las bolas según su color: azul a la izquierda, blanco en el centro y rojo a la berecha.

6.– Dado un vector $a[1..n]$ de enteros con $n \geq 1$, y dos enteros x e y, diseñar y verificar una función que sustituya en el vector a todas las apariciones de x (si las hubiese) por y.

7.– Basándose en los problemas de la relación anterior, diseñar y verificar una función que, dado un número natural n, decida si es un número "guay".

8.– Basándose también en los problemas de la relación anterior, dado un vector $a[1..n]$ de enteros con $n \geq 1$, diseñar y verificar dos funciones que decidan si el vector a es "gaspariforme" o "melchoriforme".

9.– Diseñar una versión recursiva y otra iterativa de una función que calcule la parte entera del logaritmo en base b, $b > 1$, de un número natural a. Verificar ambas versiones.

10.– Diseñar y verificar una función recursiva lineal que calcule el n-ésimo número de la sucesión de Fibonacci.