

Nro. de orden:
 LU
 Apellidos:
 Nombre:

1	2			3	4	TOTAL
	a	b	c			
25	10	5	20	20	14	97

Aclaraciones: Se permite tener UNA hoja A4 con anotaciones durante el parcial. Cualquier decisión de interpretación que se tome debe ser aclarada y justificada. Para aprobar se requieren al menos 60 puntos.

Entregar cada ejercicio en una hoja separada, numerada y que incluya el nro. de orden.

Ejercicio 1. [25 puntos] Demostrar formalmente la corrección del programa `maxInc` con respecto a su especificación usando la precondition más débil.

Especificación

```
proc maxInc (in a: Z, in b: Z, out res: Z) {
    Pre {True}
    Post {(a > b → res = a + 1) ∧ (a ≤ b → res = b + 1)}
}
```

Implementación en SmallLang

```
res := 1;
if (a < b)
    res := res + b;
else
    res := res + a;
endif
```

Ejercicio 2. [35 puntos]

Sea el siguiente ciclo con su correspondiente precondition y postcondition:

$P_c : \{i = 0 \wedge res = 0\}$

```
while (i < length(s)) do
    if (s[i] == 0)
        res := res + 1;
    else
        skip;
    endif
    i := i + 1;
endwhile
```

$Q_c : \{res = \sum_{j=0}^{\text{length}(s)-1} (\text{if } s[j] = 0 \text{ then } 1 \text{ else } 0 \text{ fi})\}$

- [10 puntos] Especificar un invariante de ciclo que permita demostrar que el ciclo cumple la postcondition.
- [5 puntos] Especificar una función variante y una cota que permitan demostrar que el ciclo termina.
- [20 puntos] Demostrar formalmente la corrección y terminación del ciclo usando el Teorema del invariante.

Ejercicio 3. [20 puntos] Dado un string¹, escribir un programa en C++ que retorne un string que indique cuántas apariciones hay de cada letra (respetando el orden del abecedario)². Por ejemplo, dada la palabra `gracias`, el programa debería devolver `a2c1g1i1r1s1`; y dada la palabra `chocolate` debería devolver `a1c2e1h1l1o2t1`.

Ejercicio 4. [20 puntos] Escribir un programa en C++ que, dada una palabra y una sopa de letras, determine si la palabra se encuentra en esta última. Diremos que una palabra pertenece a la sopa de letras si se la puede hallar en alguna de las 3 direcciones que aparecen en la imagen (considerar sólo la diagonal que va de una esquina a la otra). Suponer, además, que la grilla de letras es cuadrada.

S	E	R	V	I	S	E	R	V	I	S	E	R	V	I
K	W	X	O	K	V	A	X	O	V	W	A	X	O	V
N	A	Y	L	L	N	A	L	L	N	A	L	L	N	A
A	L	Z	I	X	A	L	I	X	A	L	Z	I	X	A
D	E	E	E	T	D	E	E	E	T	D	E	E	E	T

¹Asumir que los caracteres pueden ser únicamente las letras del abecedario, en minúscula y sin tildes.

²En caso de necesitarlo, es posible usar las funciones `char letra(int i)` que devuelve la letra ubicada en la posición `i` del abecedario y `int ord(char c)` que es la función inversa.

Ejercicio 1:

Sea Q la postcondición de la especificación.

$$\begin{aligned} wp(\text{if}(a < b) \text{ then } res := res + b \text{ else } res := res + a \text{ endif}, Q) &\equiv \\ &\equiv \underbrace{\text{def}(a < b)}_{\text{True, pues asumimos def}(a) \text{ y def}(b) \checkmark} \wedge \left[(a < b \wedge wp(res := res + b, Q)) \vee (a \geq b \wedge wp(res := res + a, Q)) \right] \equiv \\ &\equiv [a < b \wedge ((a > b) \Rightarrow res + b = a + 1 \wedge (a \leq b) \Rightarrow res = 1)] \vee \\ &\quad [a \geq b \wedge ((a > b) \Rightarrow res = 1 \wedge (a \leq b) \Rightarrow res + a = 1 + b)] \equiv \equiv E \end{aligned}$$

$$wp(res := 1, E) \equiv \text{def}(\perp) \wedge$$

$$\begin{aligned} &[(a < b \wedge (a > b) \Rightarrow b = a \wedge (a \leq b) \Rightarrow \text{True}) \vee \\ &(a \geq b \wedge (a > b) \Rightarrow \text{True} \wedge (a \leq b) \Rightarrow a = b)] \end{aligned}$$

A mí dize esto.

~~Prove that...~~

La expresión $a > b \Rightarrow b = a$ es verdadera bajo $a < b$ y también lo es $a \leq b \Rightarrow \text{True}$. ✓

Suponemos el otro caso en $a > b$ y $a = b$. Si $a > b$, $a > b \Rightarrow \text{True}$ y $a \leq b \Rightarrow a = b$ son verdaderos. Si $a = b$, $a > b \Rightarrow \text{True}$ es verdadera y $a \leq b \Rightarrow a = b$ también, por cumplirse el antecedente y el consecuente. Luego, toda la expresión es verdadera.

"—" son las expresiones que son verdaderas por ser falso el antecedente. ✓

Como la precondition más débil es True y la de la especificación también ($\text{True} \Rightarrow \text{True}$), el programa es correcto respectivamente.

16

to de su especificación - ✓

Ejercicio 2:

$$(a) I \equiv 0 \leq i \leq \text{length}(s) \wedge_{\perp} \text{res} = \sum_{j=0}^{i-1} \text{if } (S[j]=0) \text{ then } 1 \text{ else } 0 \text{ fi.} \checkmark$$

$$(b) f_v = \text{length}(s) - i \quad \text{La cota inferior que alcanza es } f_v = 0. \checkmark$$

(c) Son 5 puntos para demostrar la corrección y terminación del ciclo:

$$1. P_c \Rightarrow I?$$

Como en la precondition $i=0$, se cumple que $i \geq 0$ y que $\text{length}(s) \geq 0 = i$. Usando el valor de i , vemos que en I el rango de la sumatoria de res es vacío, entonces $\text{res} = 0$, lo que coincide con lo definido en P_c .

Luego, $P_c \Rightarrow I$. \checkmark

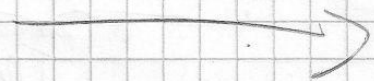
$$2. \{I \wedge B\} \text{ cuerpo } \{I\} \quad \left[\text{TOMO } B \equiv \sum_{j=0}^{i-1} \text{if } (S[j]=0) \text{ then } 1 \text{ else } 0 \text{ fi} < \text{length}(s) \right]$$

Para este cálculo la precondition más débil del ciclo y luego voy a ver si $I \wedge B \Rightarrow I$ la implica. \checkmark

$$\begin{aligned} \text{wp}(i := i + 1 \mid I) &\equiv \overline{\text{def}(i+1)} \wedge_{\perp} 0 \leq i+1 \leq \text{length}(s) \wedge_{\perp} \\ &\quad \text{res} = \sum_{j=0}^i \text{if } (S[j]=0) \text{ then } 1 \text{ else } 0 \text{ fi.} \equiv \\ &\equiv -1 \leq i < \text{length}(s) \wedge_{\perp} \text{res} = \sum_{j=0}^i \text{if } (S[j]=0) \text{ then } 1 \text{ else } 0 \text{ fi.} \end{aligned} \checkmark$$

A esta expresión lo llamo E_1 y M a la orden:

$$M = \text{if } (S[i]=0) \text{ then } 1 \text{ else } 0 \text{ fi.} \checkmark$$



148

$$wp(M, E_1) \equiv \text{def}(S[i]=0) \wedge_L \left[(S[i]=0 \wedge wp(res := res+1, E_1)) \vee (S[i] \neq 0 \wedge wp(\text{skip}, E_1)) \right] \quad \checkmark$$

Calculo aparte:

$$wp(res := res+1, E_1) \equiv \text{def}(res+1) \wedge_L \overset{\text{True pues es suma def}(res)}{-1 \leq i < \text{length}(S) \wedge_L} \\ res+1 = \sum_{j=0}^i \text{if } S[j]=0 \text{ then } 1 \text{ else } 0 \text{ fi.} \equiv \\ \equiv -1 \leq i < \text{length}(S) \wedge_L res+1 = \sum_{j=0}^i \text{if } S[j]=0 \text{ then } 1 \text{ else } 0 \text{ fi.} \quad \checkmark$$

$$wp(\text{skip}, E_1) \equiv E_1 \quad \checkmark$$

Reemplazo:

$$wp(M, E_1) \equiv 0 \leq i < \text{length}(S) \wedge_L -1 \leq i < \text{length}(S) \wedge_L \\ \left[(S[i]=0 \wedge res+1 = \sum_{j=0}^i \text{if } S[j]=0 \text{ then } 1 \text{ else } 0 \text{ fi}) \vee \right. \\ \left. (S[i] \neq 0 \wedge res = \sum_{j=0}^i \text{if } S[j]=0 \text{ then } 1 \text{ else } 0 \text{ fi}) \right] \quad \checkmark$$

Ahora voy a combinar las expresiones que se refieren al rango de i y, en el caso en que $S[i]=0$, el último término de la sumatoria es 1, que puede cancelarse con el 1 que está sumando a la variable res . En el otro caso, el último término vale 0, por lo que puede omitirse. En ambos casos res queda igualada a la misma expresión. \checkmark

$$wp(M, E_1) \equiv 0 \leq i < \text{length}(S) \wedge_L res = \sum_{j=0}^{i-1} \text{if } S[j]=0 \text{ then } 1 \text{ else } 0 \text{ fi} \wedge \\ (S[i]=0 \vee S[i] \neq 0) \quad \checkmark$$

True.

A esta expresión la llamo WPC

Nota que $I \wedge B$ es una expresión equivalente a WP_C , por lo tanto la implica. ✓

$$I \wedge B: 0 \leq i < \text{length}(s) \wedge \text{res} = \sum_{j=0}^{i-1} \text{if } S[j] = 0 \text{ then } \perp \text{ else } 0 \text{ fi} \quad \checkmark$$

$$WP_C: 0 \leq i < \text{length}(s) \wedge \text{res} = \sum_{j=0}^{i-1} \text{if } S[j] = 0 \text{ then } \perp \text{ else } 0 \text{ fi.} \quad \checkmark$$

3. $I \wedge B \Rightarrow Q_C?$

$$I \wedge B: i = \text{length}(s) \wedge \text{res} = \sum_{j=0}^{\text{length}(s)-1} \text{if } S[j] = 0 \text{ then } \perp \text{ else } 0 \text{ fi}$$

y Q_C es justamente la segunda parte de la expresión. Por lo tanto, $I \wedge B \Rightarrow Q_C$. ✓

4. $I \wedge f_v \leq 0 \Rightarrow \neg B?$

Observo que $\{f_v \leq 0\} \equiv \{\text{length}(s) - i \leq 0\}$ y eso es justamente $\neg B$. Luego, la implicación vale. ✓

5. $\{I \wedge B \wedge f_v = V_0\}$ cuando $\{f_v < V_0\}$?

$$\text{wp}(i := i + 1, f_v < V_0) \equiv \overset{\text{True}}{\text{def}} (i + 1) \wedge \text{length}(s) - i - 1 < V_0 \quad \checkmark$$

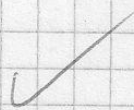
$$\text{wp}(M, \text{length}(s) - i - 1 < V_0) \equiv \text{def}(S[i] = 0) \wedge \left[(S[i] = 0 \wedge \text{wp}(\text{res} := \text{res} + 1, \text{length}(s) - i - 1 < V_0)) \vee (S[i] \neq 0 \wedge \text{length}(s) - i - 1 < V_0) \right] \quad \checkmark$$

$$\equiv (S[i] = 0 \wedge \text{length}(s) - i - 1 < V_0) \vee (S[i] \neq 0 \wedge \text{length}(s) - i - 1 < V_0) \\ \equiv \text{length}(s) - i - 1 < V_0. \quad \checkmark$$

Tomando $f_r = V_0$ de $|A| f_r = V_0$ podemos decir que
 $|A| f_r = V_0 \Rightarrow \{ \text{length}(s) - i - 1 < V_0 \}$, pues reemplazando
se obtiene ~~el~~ True.

$$\text{length}(s) - i - 1 < \text{length}(s) - i \Leftrightarrow -1 < 0.$$

Analizados los 5 puntos, puede decirse que el caso es
correcto ~~se~~ y termina.



Ejercicio 3:

```

string cuantosDeCadaLetra(string p) {
    string solucion = " ". (es vacío, no con un espacio)
    int i = 0;
    while (i < 27) { abecedario de 27 letras, o sea que empieza en 0.
        char actual = letra(i);
        int oporiciones = cantidadAporiciones(p, actual);
        if (oporiciones > 0) {
            solucion.push_back(actual);
            solucion += toString(oporiciones);
        }
        i++;
    }
    return solucion;
}

```

```

int cantidadAporiciones(string p, char e) {
    int i = 0, contador = 0;
    while (i < p.size()) {
        if (p[i] == e)
            contador++;
        i++;
    }
    return contador;
}

```

COMENTARIO:

en cuantosDeCadaLetra
no ~~contemplo~~ me molesta
el caso de p vacío porque
lo contemplo en
cantidadAporiciones; p

Ejercicio 4

```

bool estaEnSopa (string p, vector<vector<char>> sopa) {
    bool enFila = false;
    bool enColumna = false;
    bool enDiagonal = false;
    int i = 0;
    if (sopa.size() > 0) {
        while (i < sopa.size() && !estaEnFila(p, sopa, i)) {
            i++;
        }
        enFila = (i < sopa.size());
        i = 0;
        while (i < sopa[0].size() && !estaEnColumna(p, sopa, i)) {
            i++;
        }
        enColumna = (i < sopa[0].size());
        enDiagonal = estaEnDiagonal(p, sopa);
    }
    return (enFila || enColumna || enDiagonal);
}

```

```

bool estaEnFilas (string p, vector<vector<char>> sopa, int f) {
    int i=0, j=0;
    while (i < sopa[0].size() && j < p.size()) {
        if (sopa[f][i] == p[j]) {
            j++;
        } else {
            j=0;
        }
        i++;
    }
    return j == p.size();
}

```

sopa[f] = ababc.

P = abc.

```

bool estaEnColumnas (string p, vector<vector<char>> sopa, int c) {
    int i=0, j=0;
    while (i < sopa.size() && j < p.size()) {
        if (sopa[i][c] == p[j]) {
            j++;
        } else {
            j=0;
        }
        i++;
    }
    return j == p.size();
}

```

```
bool estaEnDiagonal (string p, vector<vector<char>> sopa) {  
    int i=0, j=0;  
    while (i < sopa.size() && j < p.size()) {  
        if (sopa[i][i] == p[j]) {  
            j++;  
        } else {  
            j=0;  
            i++;  
        }  
    }  
    return j == p.size();  
}
```

Muy Prolija!

