UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación– Algoritmos y Estructura de Datos I
Segundo parcial – 30/06/2017

1	2			3	4	TOTAL
	a	b	c			
25	10	5	20	20	17	97

Nro. de orden:
LU
Apellidos:
Nombres

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 precondición 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 precondición y postcondición:

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

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

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

a) **[10 puntos]** Especificar un invariante de ciclo que permita demostrar que el ciclo cumple la postcondición.

b) **[5 puntos]** Especificar una función variante y una cota que permitan demostrar que el ciclo termina.

c) **[20 puntos]** Demostrar formalmente la corrección y terminación del ciclo usando el Teorema del invariante.

Ejercicio 3. [20 puntos] Dado un string[1], 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.

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

$wp(\text{if } (a<b) \text{ then } res := res+b \text{ else } res := res+a \text{ endif}, Q) \equiv$

$\equiv \underbrace{def(a<b)}_{\text{True, pues asumo } def(a) \text{ y } def(b)} \wedge_L [(a<b \wedge wp(res := res+b, Q)) \vee (a \geq b \wedge wp(res := res+a, Q))] \equiv$

$\equiv [a<b \wedge ((a>b) \Rightarrow res+b = a+1 \wedge (a \leq b) \Rightarrow res = 1)] \vee$

$[a \geq b \wedge ((a>b) \Rightarrow res = 1 \wedge (a \leq b) \Rightarrow resta = 1+b)] \equiv E$

$wp(res := 1, E) \equiv def(1) \wedge_L$

$[((a<b \wedge (a>b)) \Rightarrow b = a \wedge (a \leq b) \Rightarrow \text{True}) \vee$

$(a \geq b \wedge (a>b) \Rightarrow \text{True} \wedge (a \leq b) \Rightarrow a = b)]$

Analizo esto.

~~Bajo el caso a<b,~~

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

Separemos 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, todo la expresión es verdadera.

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

Como la precondición más débil es True y la de la especificación también (True $\Rightarrow$ True), el programa es correcto respec-

to de su especificación - ✓

Ejercicio 2:

(a) $I \equiv 0 \leq i \leq length(s) \wedge_L res = \sum_{j=0}^{i-1} if\ (s[j]=0)\ then\ 1\ else\ 0\ fi$ ✓

(b) $f_v = length(s) - i$ La cota inferior que alcanza es $f_v = 0$. ✓

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

[1] $P_c \Rightarrow I$?

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

[2] $\{I \wedge B\}$ cuerpo $\{I\}$? [TOMO $B \equiv \{i < length(s)\}$]

Para esto calculo la precondición más débil del ciclo y luego voy a ver si $I \wedge B \Rightarrow$ la implica. ✓

$wp(i := i+1)I) \equiv$ TRUE, pues asumo $def(i)$ ✓

$def(i+1) \wedge_L 0 \leq i+1 \leq length(s) \wedge_L$

$res = \sum_{j=0}^{i} if\ s[j]=0\ then\ 1\ else\ 0\ fi. \equiv$

$\equiv -1 \leq i < length(s) \wedge_L res = \sum_{j=0}^{i} if\ s[j]=0\ then\ 1\ else\ 0\ fi.$ ✓

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

$M = if\ (s[i]=0)\ then\ 1\ else\ 0\ fi.$ ✓

→

1d

$$wp(M, E_1) \equiv def(S[i]=0) \wedge_L \Big[(S[i]=0 \wedge wp(res := res+1, E_1)) \vee (S[i] \neq 0 \wedge wp(skip, E_1)) \Big]$$ ✓

Calculo aparte:

$wp(res := res+1, E_1) \equiv$ def(res+1) true pues asumo def(res) $\wedge_L -1 \leq i < length(S) \wedge_L$

$$res + 1 = \sum_{j=0}^{i} \text{if } S[j]=0 \text{ then } 1 \text{ else } 0 \text{ fi} \equiv$$

$$\equiv -1 \leq i < length(S) \wedge_L res+1 = \sum_{j=0}^{i} \text{if } S[j]=0 \text{ then } 1 \text{ else } 0 \text{ fi}$$ ✓

$wp(skip, E_1) \equiv E_1$ ✓

Reemplazo:

$$wp(M, E_1) \equiv 0 \leq i < length(S) \wedge_L -1 \leq i < length(S) \wedge_L$$

$$\Big[(S[i]=0 \wedge res+1 = \sum_{j=0}^{i} \text{if } S[j]=0 \text{ then } 1 \text{ else } 0 \text{ fi}) \vee (S[i] \neq 0 \wedge res = \sum_{j=0}^{i} \text{if } S[j]=0 \text{ then } 1 \text{ else } 0 \text{ fi}) \Big]$$

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 combinarse 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. ✓

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

(TRUE)

A esta expresión la llamo WP_c

Noto que $I \wedge B$ es una expresión equivalente a WP_c, por lo tanto lo implica. ✓

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

WP_c: $0 \leq i < length(s) \wedge_L res = \sum_{j=0}^{i-1} if\ s[j]=0\ then\ 1\ else\ 0\ fi$. ✓

[3] $\therefore_c$ $I \wedge \neg B \Rightarrow Q_c$?

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

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

[4] $\therefore_c$ $I \wedge f_v \leq 0 \Rightarrow \neg B$?

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

[5] $\therefore_c$ $\{I \wedge B \wedge f_v = V_0\}$ cuerpo $\{f_v < V_0\}$?

$wp(i := i+1, f_v < V_0) \equiv \overbrace{def(i+1)}^{true} \wedge_L length(s) - i - 1 < V_0$ ✓

$wp(M, length(s) - i - 1 < V_0) \equiv def(s[i]=0) \wedge_L [(s[i]=0 \wedge wp(res := res+1, length(s)-i-1<V_0)) \vee$

$(s[i] \neq 0 \wedge length(s) - i - 1 < V_0)] \equiv$ ✓

$\equiv (s[i]=0 \wedge length(s) - i - 1 < V_0) \vee (s[i] \neq 0 \wedge length(s) - i - 1 < V_0)$

$\equiv length(s) - i - 1 < V_0$. ✓

Tomando $f_v = V_0$ de $I \wedge B \wedge f_v = V_0$ podemos decir que $I \wedge B \wedge f_v = V_0 \Rightarrow \{length(s) - i - 1 < V_0\}$, pues reemplazando se obtiene ~~el~~ True.

$$length(s) - i - 1 < length(s) - i \iff -1 < 0 .$$

Analizados los 5 puntos, puede decirse que el ciclo es correcto ~~sea~~ y termina.

Ejercicio 3:

```
string cuantosDeCadaLetra(string p){
    string solucion = " ".   (es vacío, no con un espacio)
    int i=0;
    while(i<27){   descendo de 27 letras, asumo que empieza en 0.
        char actual = letra(i);
        int aparaciones = cantidadAparaciones(p, actual);
        if(aparaciones>0){
            solucion.push_back(actual);
            solucion += toString(aparaciones);
        }
        i++;
    }
    return solución;
}
int cantidadAparaciones(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 cantidadAparaciones;

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() && !estaEnFilas(p, sopa, i)){
            i++;
        }
        enFila = (i < sopa.size());
        i = 0;
        while (i < sopa[0].size() && !estaEnColumnas(p, sopa, i)){
            i++;
        }
        enColumna = (i < sopa[0].size());
        enDiagonal = estaEnDiagonal(p, sopa);
    }
    return (enFila || enColumna || enDiagonal);
}
```

sigo atrás →

```
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();
}

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();
}
```

sopa[f] = abcabc.

P = abc.

```
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 Bonito!