Ejercicio 1. a) Reescribamos $post\{0 \leq result \leq |l| - 1 \wedge_L l[result] = elem\}$

b) Está mal , por que con i=0 , falla dado que s[-1] no está definido. modificando el rango para que i sea mayor (no mayor o igual) que cero, ya es suficiente. Ni siquiera tendría que ver en la pre que mi lista no sea vacía , por que si fuera vacia no habría nada entre 1 y 0 por lo tanto siempre seria falsa ese rango entonces siempre true esa afirmacion entera y todo la posta seria true.

c) Reescribo la post $post\ \{result \in l \wedge (\forall y \in \mathbb{Z})((y \in l \wedge y \neq result) \rightarrow y > result)\}$

Ejercicio 2. a) $l =< 6, 7, 8 > suma = 1$ la subsequencia $< 6 >$ no suma 1

b) Seguiría siendo incorrecto , con el mismo ejemplo min suma es 0 , dado que no hay ningun negativo en la lista y max suma seria 21 , por lo tanto suma podría ser 1 como antes

c) Reescribamos $Pre\{(\exists s \in seq < \mathbb{Z} >)(esSubSecuencia(s, l) \wedge suma = \sum_{i=0}^{|s|-1} s[i])\}$

$pred\ esSubSecuencia(s : seq < \mathbb{Z} >, l : seq < \mathbb{Z} >)\{(\forall x \in \mathbb{Z})(x \in s \rightarrow \#apariciones(x, s) \leq \#apariciones(x, l))\}$

Ejercicio 3. a) I) Posibles = 0

II) Posibles = {1,-1}

III) Posibles = $\{\sqrt{27}, -\sqrt{27}\}$

b) I) 3

II) 0 o 3

III) Cualquier indice válido

c) I) 3

II) 0

III) 0

d) Cuando hay una sola aparición del máximo

Ejercicio 4. a) Esta mal por el y luego , cualquier a va a ser o mayor o igual que cero o menor que cero , por lo tanto no va a cumplir ambas guardas nunca

b) Pusieron un o , eso mejoró , pero falto el menor o igual para a mayor que 0

c) Está bien

d) Esá bien

e) el o causa problemas , supongamos que ponemos un result diferente de 2.b y un a menor que 0 entonces la primera condicion no se cumple por que seria true implicando false, entonces se tiene que cumplir la segunda como a ¡ 0 automaticamente la segunda se cumple , por lo que result podria ser cualquier cosa que no sea 2.b

f) Parece correcta por lo menos conceptualmente , no se si es válido escribirlo así

Ejercicio 5. a) Devuelve 9 , si , cumpele la postcondición

b) Con 0,5 no funciona , por que el cuadrado de 0,5 es mas pequeño que 0,5 . Con el 0,2 funciona por que lo hace NO negativo , lo mismo con el -7 y finalmente con el 1 no funciona por que 1 al cuadrado es 1

c) $pre\{a > 1 \vee a < 0\}$

Ejercicio 6. a) $P3 \rightarrow P1 \rightarrow P2$

b) $Q3 \rightarrow Q1 \rightarrow Q2$ (En número reales)

c) No se si se refiere a escribir en algun lenguaje , o en small talk de ambas formas es trivial

d) I) Si , por que $x \leq -10 \rightarrow x \leq 0$

II) No , por que $x \leq 10$ no implica $x \leq 0$ entonces el algoritmo, basado en E1 podria hacer cosas raras como enviar cualquier x mayor que cero a 0, cumpliendo E1 , pero no cumpliendo esta especificacion con por ejemplo x = 4

III) Si por que la post de E1 implica la post de este ejemplo

IV) No por que la post de E1 no implica la posta implica la posta de este ejemplo

V) Si porque tanto la pre como la post de E1 implican la pre y post de este ejemplo

VI) No se cumple la pre de E1 , entonces puede suceder cualquier cosa en esos casos para la post, cosas que no cumpla la post como por ejemplo si el algoritmo enviaba todos los numeros positivos menores que 10 al -20 y el resto cumple E1 , en ese caso se cumple E1 , pero en este ejemplo el 4 cumple la pre sin embargo el algoritmo devolvio algo negativo entonces no cumple la post del ejemplo

VII) La post de E1 no implica la post de este ejemplo entonces puedo tener casos que cumple la post de E1 , pero no cumple la post de este ej que es mas restringida VIII) Devuelta la pre del E1 no implica la del ejemplo , por lo tanto con los casos que no cumplen en E1 podria hacer cualquier cosa el algortimo , y esos casos cumpliendo el ejemplo podrian no cumplir la post

e) La precondición de mi reemplazo debe ser implicada por la condición del original

La postcondición de mi reemplazo debe ser implicada por la condicion del original

Ejercicio 7. a) si se cumple pre de p1 entonces $x \neq 0$ por lo tanto $n \leq 0 \rightarrow x \neq 0$ es lo mismo que $n \leq 0 \rightarrow True$ que es siempre verdadera

b) Si , por que son equivalentes las post

c) Esto es cierto , por que toda la especificacion de P1 , implica la de P2 , por lo tanto podemos decir que P1, esta contenida en P2 , como nosotros hicimos algoritmo en base a P2 , entre comillas consideramos todos los casos de P1 y alguno más, pero esos no nos importan por que P1 no los contempla

Ejercicio 8. la post de n-esimo1 no implica la de n-esimo2 y viceversa , por lo tanto , un algoritmo que cumple un proc no necesariamente cumple el del otro

Ejercicio 9. a) *proc* esPar (in $z : \mathbb{Z}$){

$pre\{True\}$

$post\{result = True \iff z \bmod 2 = 0\}$

}

b)*in* n,m: $\mathbb{Z}$, *out* result: *Bool*

$pre\{True\}$

$post\{result = True \iff (\exists k \in \mathbb{Z})(n = mk)\}$

c) $pre\{\frac{n}{m} \neq 0\}$

$post\{result = \frac{m}{n}\}$

d) in l lista out result lista $pre\{True\}$

$post\{(\forall j \in \mathbb{N})(0 \leq j < |l| \rightarrow_L (ord('0') \leq ord(l[j]) \leq ord('9')) \rightarrow l[j] \in result)\}$

e)$pre\{True\}$

$post\{|result| = |l| \wedge (\forall i \in \mathbb{Z})(((0 \leq i < |s| \wedge i \bmod 2 \neq 0) \rightarrow result[i] = 2 * l[i])$

$\wedge(0 \leq i < |s| \wedge_L i \bmod 2 = 0 \rightarrow result[i] = l[i])\}$

f) in z entero , out result seq $pre\{True\}$

$post\{(\forall n \in \mathbb{N})(z \bmod n = 0 \rightarrow (n \in result \wedge \#apariciones(n, result) = 1)) \wedge (z \bmod n \neq 0 \rightarrow n \notin result)\}$

Ejercicio 10. a) No , no tiene sentido , por que cero no tiene multiplos dado que multiplicar algo por cero siempre da cero

b) No debería serlo por a) y no esta especificación no permite m=0 como entrada vease la precondición

c) $pre\{True\}$ $post\{result = (\neg(m = 0) \wedge_L (n \bmod m = 0))\}$

d) True es la afirmación mas débil que hay , asi que la anteriór precondición implica la nueva. En otra palabras la nueva precondición es mas débil que la anteriór. Lo cual tiene sentido , por que la anteriór no dejaba que pase un m=0 , pero esta si.

Ejercicio 11. a) Todo indica que no , por que devolver la que resulta de duplicar sus valores en posiciones pares esta insinuando que sus valores pares deberíab quedar iguales

b) Si, la satisface, por que la post no pide nada con respecto a las posiciones pares

c) habria que agregar un and y hacer lo mismo solo que cambiando mod 2 = 0 entonces $result[i] = l[i]$

d) La nueva post es mas fuerte, dado que implica la anteriór post, en otras palabras agrega una restricción más, pero la anteriór restricción la sigue cumpliendo , en otras palabras cumple la post anteriór y además cumple otras cosas más.

Ejercicio 12. $procintToBinary$(in $z : \mathbb{Z}$,out $l : seq < \{0, 1\} >$){
$pre\{0 \leq z\}$
$post\{soloHay1y0s(l) \wedge (\forall x \in l)(z = \sum_{i=0}^{|l|-1} l[i].2^i)\}$
}
$pred$ $soloHay1y0s$(in $result : seq < \mathbb{Z} >$){
$(\forall elem \in \mathbb{Z})(elem \in result \rightarrow ((elem = 0) \vee (elem = 1))$
}

Ejercicio 13. No por que al no tener referencia sobre que se quería lograr con la especificación no puede haber sobre o sub especificación

Ejercicio 14. a) $proc$ $sumarFactoresPrimos$(in $z : \mathbb{Z}$,out : $result : \mathbb{Z}$){
$pre\{z \geq 0\}$
$post\{result = \sum_{i=0}^{z}$ if $esPrimo(r) \wedge (z \bmod 2 = 0)$ then i else $0\}$
}
b) $proc$ $esPerfecto$(in $z : \mathbb{Z}$,out : $result : Bool$){
$pre\{z \geq 0\}$
$post\{result = True \iff z = \sum_{i=0}^{z-1}$ if z mod i = 0 then i else $0\}$
}
c) $proc$ $menorEnteroPositivo$(in $z : \mathbb{Z}$,out : $result : \mathbb{Z}$){
$pre\{z \geq 0\}$
$post\{result > 1 \wedge esCoprimo(z, result) \wedge (\forall r \in \mathbb{Z})(esCoprimo(r, z) \rightarrow result \leq r)\}$
}
Es coprimo esta hecho en la guía anteriór.
d) Hecho en clase.

e) *proc* $difEntreMaxyMin$(in $s : Seq < \mathbb{Z} >$,out : $result : \mathbb{Z}$){
$pre\{True\}$
$post\{(\forall x, y \in \mathbb{Z})(0 \leq x, y < |s| \rightarrow result = s[x] - s[y]\}$
}
f) *proc* $devolverDivisor$(in $s : Seq < \mathbb{Z} >$,out : $result : \mathbb{Z}$){
$pre\{True\}$
$post\{result \in s \wedge (\exists i \in \mathbb{Z})(0 \leq i < |s| \rightarrow s[i] \bmod result = 0) \wedge$
$(\forall i \in \mathbb{Z})(0 \leq i, r < |s| \rightarrow (\sum_{j=0}^{|s|-1}$if $s[j] \bmod result = 0$ then 1 else 0
$\geq \sum_{j=0}^{|s|-1}$ if $s[j] \bmod s[i] = 0$ then 1 else 0))

Ejercicio 15. a) *proc* $nEsimaAparicion$(in $s : seq < \mathbb{R} >$, in: $e : \mathbb{R}$, in: $n : \mathbb{Z}$, out result: $\mathbb{Z}$){
$pre\{\#apariciones(e, s) = n \geq 1\}$
$post\{(l[result] = e) \wedge (n = \sum_{i=0}^{result}$ if $l[result] = e$ then 1 else 0$\})$
}
b) *proc* $nEsimaAparicion$(in : $seq < \mathbb{R} >$, in: $e : \mathbb{R}$, in: $n : \mathbb{Z}$, out result: $\mathbb{Z}$){
$pre\{|l| \neq 0 \wedge e \in seq < \mathbb{R} >\}$
$post\{(l[result] = e) \wedge (n = \sum_{i=0}^{result}$ if $l[result] = e$ then 1 else 0$\})$
}
c) *proc* $incluida$(in lista1:$seq < \mathbb{Z} >$, in lista2: $seq < \mathbb{Z} >$,out result: $Bool$){
$pre\{|s| \leq |t|\}$
$post\{result = True \iff \neg(\exists i \in \mathbb{Z})(0 \leq i < |lista1| \rightarrow lista1[i] \notin lista2) \wedge (\forall i \in \mathbb{Z})(0 \leq i < |lista1| \rightarrow \#apariciones(lista1[i], lista1) \leq \#apariciones(lista1[i], lista2))\}$
}
d) *proc* $mezclarOrdenado$(in s,t:$seq < \mathbb{Z} >$,out result: $resultl$){
$pre\{|s| \neq |t|\}$
$post\{(\forall i \in \mathbb{Z})(0 \leq i < |result| \rightarrow_L result[i] \in s \vee result[i] \in t) \wedge estaOrdenada(result) \wedge |result| = |s| + |t|\}$
}
e) *proc* $interseccionSinRepetidos$(in s,t:$seq < \mathbb{R} >$,out result: $seq < \mathbb{R} >$){
$pre\{True\}$
$post\{(\forall j \in \mathbb{Z})(0 \leq j < |result| \rightarrow_L result[j] \in s \wedge result[i] \in t)\}$
}
f) *proc* $interseccion$ (in s,t:$seq < \mathbb{R} >$,out result: $seq < \mathbb{R} >$){
$pre\{True\}$
$post\{(\forall j \in \mathbb{Z})(0 \leq j < |result| \rightarrow_L result[j] \in s \wedge result[i] \wedge$
$\#apariciones(result[j], result) = 1 \in t)\}$
}

Ejercicio 16. a) *proc* $cantidadApariciones$ (in $l : seq < Char >$,out $result : seq < Char \times \mathbb{Z} >$){
$pre\{True\}$
$post\{(\forall j \in \mathbb{Z})(0 \leq j < |result| \rightarrow result[j][0] \in l \wedge \#apariciones(result[j][0], l) = result[j][1]\}$
}

b) *proc devolvePrefijos* (in $l : seq < \mathbb{Z} >$,out $result : seq < seq < \mathbb{Z} >>$){
$pre\{True\}$
$post\{(\forall j \in \mathbb{Z})(0 \leq j < |result| - 1 \rightarrow |result[j]| = |result[j+1]| - 1) \wedge$
$(\forall k \in \mathbb{Z})(0 \leq k < |result| \rightarrow_L result[k] = subseq(l, 0, k+1))\}$
}
c) *proc s* (in $l : seq < seq < \mathbb{Z} >>$,out $result :< seq < \mathbb{Z} >>$){
$pre\{True\}$
$post\{(\forall j \in \mathbb{Z})(0 \leq j < |l| \rightarrow_L maximo(l[j]) \leq maximo(result)) \wedge result \in l\}$
}
aux maximo (in : $seq < \mathbb{Z} >$)$\mathbb{Z} = \sum_{i=0}^{|in|-1} if(\forall j \in \mathbb{Z})(0 \leq j < |in| \wedge \rightarrow_L l[i] \geq l[j]$ then $\frac{l[i]}{\#apariciones(l[i],l)}$ else 0)
d) *proc interseccionMultiple* (in $l : seq < seq < \mathbb{R} >>$,out $result :< seq < \mathbb{R} >>$){
$pre\{True\}$
$post\{(\forall j \in \mathbb{Z})(0 \leq j < |result| \rightarrow_L ((\forall i \in \mathbb{Z})(0 \leq i < |l| \rightarrow_l result[j] \in l[i]))\}$
}
e) *proc le* (in $l : seq < \mathbb{Z} >$,out $result :< seq <$seq $\mathbb{Z} >>$){
$pre\{True\}$
$post\{|result| = 2^{|l|} \wedge (\forall j \in \mathbb{Z})(0 \leq j < |result| \rightarrow_L esSubsecuencia(result[j], l) \wedge (\forall i \in \mathbb{Z})(0 \leq i < |result[j]| \rightarrow_L result[j][i] = l[i] \wedge (\forall j \in \mathbb{Z})(0 \leq j < |result| \rightarrow_L (\forall i \in \mathbb{Z})(0 \leq i < |result| \wedge i \neq j \rightarrow_L result[j] \neq result[i]))\}$
}
Casi seguro que la condicion del medio no es necesaria, con pedir que sean todas subsecuencias , que no haya repetida y que haya $2^{|conjunto|}$, seguro voy a tener partes del conjunto

Ejercicio 17. a) Esta proceso devuelve 3 cosas
b) este proceso no devuelve nada
c) es la correcta
d) parece correcta , devuelve la suma

Ejercicio 18. a) Correcta
b) Correcta
c) Incorrecta se indefine la post, por que L0 no está definida
d) Correcta , si bien modifica l , el resultado es corecto
e) Esta post va a ser siempre falsa por que $|l| \neq |L_0| - 1 = |l| - 1$

Ejercicio 19. a) La especificación es válida, describe la acción de intercambiar dos valores de una lista
b) Por ejemplo si ingresamos $< 1, 2, 3 >, 0, 1$ debería cambiar el primer valor con el segundo , sin embargo una salida correcta acorde a la especificación es $< 2, 1, 4 >$. Cambió lo que tenía que cambiar pero ademas cambió el último valór , como mantiene la misma longitud está bien.
c)Agregaría $(\forall r \in \mathbb{Z})(0 \leq r < |l| \wedge r \neq i, j \rightarrow_L l[i] = L_0[i])$

Ejercicio 20. Este especificación recibe una lista y una posición , en la lista en esa posición pone lo que había primero en la lista y en el resto de los lugares pone lo que había en la

posición , y ademas modifica la posición recibida por el valor que había inicialmente en la lista en la posición

Ejercicio 21. a) Le falta indicar que en las posiciones impares se mantengan los valores
$(\forall j \in \mathbb{Z})(0 \leq j < |l| \wedge i \bmod 2 \neq 0 \rightarrow_L l[i] = L_0[i])$