Integral Types

Nom i Cognoms: Xavier Berga Puig
DNI: 48257114D
Grup de pràctiques: GM1
Data: 07/03/2017

ÍNDEX:

1. Descripció de les funcions implementades

1.1 Apartat 1

En el primer apartat ens demanen implementar una funció booleana anomenada **allBits**, aquesta ha de comprovar si tots els elements d'un vector d'enters són 0 o 1.
El paràmetre que ve amb la funció és: num (un vector d'enters).

- He fet servir un bucle for que vagi iterant fins a recórrer totes les posicions del vector comprovant així cada element.
    - Si un d'ells és diferent de 0 o 1 retorna false i surt de la funció.
    - Si ha recorregut tot el for sense haver-se trobat amb cap element diferent de 0 o 1, retornarà true.

1.2 Apartat 2

En aquest apartat ens demanen implementar una funció anomenada **copy**, la funcionalitat de la qual és copiar els elements de la cadena de caràcters from al vector d'enters to amb una sèrie de requisits:

-La funció deixarà de copiar tant si s'ha acabat la cadena com si no queden més posicions lliures al vector.
-Solament copiarem els bits 0 o 1, si hi ha algun caràcter diferent, l'ignorarem.
-En el vector, la posició 0 es correspon al bit de menor pes, en la cadena en canvi es correspon amb el bit de major pes.

Els paràmetres que venen amb la funció són: from (una cadena de caràcters) i to (un vector d'enters).

- He declarat una variable entera inicialitzada a 0 anomenada pos, on emmagatzemaré la posició del vector to en cada moment.

- Tot seguit, he fet un bucle for que recorri el vector de caràcters from al revés, és a dir, des de l'última posició (from.length-1), fins a la posició 0 inclosa.

- En cada iteració d'aquest bucle, entraré dins del if en cas que l'element actual del vector from sigui un 0 o bé un 1, (ho comprovo fent un from.charAt(...) – '0').
  L'hi resto el caràcter 0 per a saber el valor enter que conté l'element del String.

    - En cas que es compleixi la condició anterior, guardaré l'element llegit del vector de caràcters from a la posició pos del vector to i incrementaré el comptador de posicions.
    - En cas contrari, si es tracta d'un caràcter diferent de 0 i 1, faré la següent iteració del bucle for sense incrementar la variable pos, ja que esperaré a guardar-hi el següent 0 o 1 que trobi.

1.3 Apartat 3

En aquesta funció entera anomenada **narrow**, ens demanen implementar la conversió d'un vector anomenat from, a un altre vector amb un nombre de bits inferior o igual a l'original.
Els paràmetres que venen amb la funció són: from (un vector d'enters) i toLength (la mida del nou vector).
Ens garanteixen la precondició següent: 0 <= toLength <= from.length

- Primerament he declarat un vector d'enters anomenat resultat amb mida toLength. Com em diuen que la mida és més petita o igual que l'original, no em cal preocupar per l'extensió de signe.
- Per a copiar el vector, he fet un bucle for amb toLength iteracions que recorri el vector resultat.
    - A cada iteració guardo el valor de la posició actual del vector from a la mateixa posició del vector resultat.

1.4 Apartat 4

En aquesta funció entera anomenada **widen**, ens demanen tractar el cas oposat de l'apartat anterior, haurem de fer la conversió a un nombre major de bits. Per aquest motiu, en aquest apartat sí que haurem de tenir en compte l'extensió de signe segons el bit de la posició from.length del vector from.
Els paràmetres que venen amb la funció són: from (un vector d'enters) i toLength (la mida del nou vector).
Ens garanteixen la precondició: 0 <= from.length <= toLength

- Primerament, he declarat un vector d'enters anomenat resultat amb mida toLength.
- Després fet un bucle for amb tantes iteracions com elements tingui el vector from.
    - A cada iteració guardo el valor de la posició actual del vector from a la mateixa posició del vector resultat.
- Un cop acabat el for, tinc el vector from copiat al vector resultat en les posicions menys significatives, però això no és tot, ja que totes les posicions excedides del tamany del vector from, tindran 0's, per tant, haurem de fer l'extensió de signe si es correspon.
- Per a comprovar si cal fer l'extensió de signe, he fet un if que miri si el bit que hi ha a la posició més significativa del vector from (from.length) és un 1.
    - En cas que així sigui, entro al if i faig l'extensió de signe mitjançant un bucle for que va des de l'última posició del vector from (from.length) fins a l'última posició del vector resultat (toLength), i per a cada iteració, he posat que l'element del vector resultat en aquella posició valgui 1.
    - En cas contrari retorno el vector resultat, ja que estava inicialitzat a 0 i per tant les posicions no modificades ja contenen 0's.

1.5 Apartat 5

En aquesta funció entera anomenada **cast**, ens demanen que tractem els casos depenent del tamany del vector from (from.length) i el tamany desitjat (toLength), i fem una crida a les funcions **narrow** o **widen** implementades anteriorment segons correspongui.
Els paràmetres que venen amb la funció són: from (un vector d'enters) i toLength (la mida desitjada).

- Per a implementar-la, he utilitzat una estructura if/else:
    - En cas que la mida del vector donat sigui més gran o igual que la mida desitjada entro al if i faig un return de la funció narrow amb els paràmetres corresponents.
    - En cas contrari, faig un return de la funció widen amb els paràmetres corresponents.

1.6 Apartat 6

En aquest apartat ens demanen implementar una funció anomenada **and**, que calculi bit a bit l'operació lògica and de dos vectors d'enters. En cas que els dos paràmetres que ens passen es puguin representar amb 32 bits, aquests els transformo a mida Integer (32 bits), en cas contrari, els transformo a mida Long (64 bits).
Els paràmetres que venen amb la funció són: arg1 (un vector d'enters) i arg2 (un segon vector d'enters).

- Primerament he utilitzat una estructura if/else per a actualitzar els vectors arg1 i arg2:
    - Si la mida del vector arg1 és més petita o igual a 32, transformo arg1 i arg2 com a vectors de mida 32 amb extensió de signe mitjançant la funció toInteger().
    - En cas que la mida del vector arg1 sigui superior a 32, transformo arg1 i arg2 a vectors de mida 64 amb extensió de signe mitjançant la funció toLong().
- Un cop tinc els dos vectors expressats amb 32 o 64 bits segons correspongui, creo un nou vector amb la nova mida d'arg1 anomenat resultat.
- Després, mitjançant un bucle for, recorro posició a posició els dos vectors arg1 i arg2 i el vector resultat.
    - Utilitzant una funció que he implementat a part anomenada operacio_and, vaig fent les operacions dels elements que anem recorrent i els vaig guardant a les respectives posicions del vector resultat.
    - La funció operacio_and, calcula la and lògica donats dos valors, si els dos valors són 1, retorna 1, en cas contrari, retornarà 0. Els dos valors que l'hi passem són els elements d'arg1 i arg2.

1.7 Apartat 7

En aquest apartat ens demanen implementar una funció anomenada **or**, que calculi bit a bit l'operació lògica or de dos vectors d'enters. En cas que els dos paràmetres que ens passen es puguin representar amb 32 bits, aquests els transformo a mida Integer (32 bits), en cas contrari, els transformo a mida Long (64 bits).
Els paràmetres que venen amb la funció són: arg1 (un vector d'enters) i arg2 (un segon vector d'enters).

- Primerament he utilitzat una estructura if/else per a actualitzar els vectors arg1 i arg2:
    - Si la mida del vector arg1 és més petita o igual a 32, transformo arg1 i arg2 com a vectors de mida 32 amb extensió de signe mitjançant la funció toInteger().
    - En cas que la mida del vector arg1 sigui superior a 32, transformo arg1 i arg2 a vectors de mida 64 amb extensió de signe mitjançant la funció toLong().
- Un cop tinc els dos vectors expressats amb 32 o 64 bits segons correspongui, creo un nou vector amb la nova mida d'arg1 anomenat resultat.
- Després, mitjançant un bucle for, recorro posició a posició els dos vectors arg1 i arg2 i el vector resultat.
    - Utilitzant una funció que he implementat a part anomenada operacio_or, vaig fent les operacions dels elements que anem recorrent i els vaig guardant a les respectives posicions del vector resultat.
    - La funció operacio_or, calcula l'or lògica donats dos valors, si els dos valors són 0, retorna 0, en cas contrari, retornarà 1. Els dos valors que l'hi passem són els elements d'arg1 i arg2.

1.8 Apartat 8

En aquest apartat ens demanen implementar una funció anomenada **leftShift**, aquesta, desplaça tantes posicions a l'esquerra el vector num com el valor enter que tingui numPos.
Els paràmetres que venen amb la funció són: num (un vector d'enters) i numPos (un enter).

- Primerament he utilitzat una estructura if/else per a actualitzar el vector num:
    - Si la mida del vector num és més petita o igual a 32, el transformo com a un vector de mida 32 amb extensió de signe mitjançant la funció toInteger().
    - En cas que la mida del vector num sigui superior a 32, el transformo com a un vector de mida 64 amb extensió de signe mitjançant la funció toLong().
- Depenent de la nova mida del vector num, també transformo el valor de numPos utilitzant l'operació % perquè quedi un nombre mòdul amb la mida vàlid i pugui fer així el desplaçament correctament.

- Un cop tinc el vector transformat a la mida corresponent, creo un vector anomenat resultat amb la nova mida del vector num.
- Després inicialitzo una variable a 0 anomenada pos que em servirà per a recórrer les posicions del vector num.
- Tot seguit, faig un bucle for des de la posició numPos fins a la llargada del vector num, ja que de la posició 0 fins a la numPos, haurem de deixar els 0 resultat d'haver desplaçat el vector.
    - A cada iteració del bucle, guardo al vector resultat el valor que hi ha a la posició pos del vector num i incremento la variable pos.

1.9 Apartat 9

En aquest apartat, ens demanen implementar una funció anomenada **unsignedRightShift** que faci el mateix que en l'apartat anterior, però aquest cop desplaçant els bits cap a la dreta.
D'igual forma, els paràmetres que ens donen són: num (un vector d'enters) i numPos (el nombre de posicions a desplaçar).

- Primerament he utilitzat una estructura if/else per a actualitzar el vector num:
    - Si la mida del vector num és més petita o igual a 32, el transformo com a un vector de mida 32 amb extensió de signe mitjançant la funció toInteger().
    - En cas que la mida del vector num sigui superior a 32, el transformo com a un vector de mida 64 amb extensió de signe mitjançant la funció toLong().
- Depenent de la nova mida del vector num, també transformo el valor de numPos utilitzant l'operació % perquè quedi un nombre mòdul amb la mida vàlid i pugui fer així el desplaçament correctament.
- Un cop tinc el vector transformat a la mida corresponent, creo un vector anomenat resultat amb la nova mida del vector num.
- Després, creo una variable anomenada pos inicialitzada amb la mida del vector num, aquesta ens servirà per a recórrer el vector num al revés.
- Tot seguit, amb un bucle for que vagi disminuint des de la posició [num.length – numPos] fins a 0, recorrerem el vector resultat.
    - A cada iteració del bucle, copiarem el valor que hi ha a la posició pos del vector num, al vector resultat, i restarem 1 la variable pos.
- Les posicions des de numPos fins a num.length no les tocarem, ja que havien quedat inicialitzades a 0 i corresponen al nombre de posicions que hem de desplaçar.

1.10 Apartat 10

Finalment, en aquest apartat ens demanen implementar una funció anomenada **signedRightShift** que faci el mateix que la funció anterior però fent extensió de signe en cas de ser necessari.
D'igual forma, els paràmetres que ens donen són: num (un vector d'enters) i numPos (el nombre de posicions a desplaçar).

- Primerament he utilitzat una estructura if/else per a actualitzar el vector num:
    - Si la mida del vector num és més petita o igual a 32, el transformo com a un vector de mida 32 amb extensió de signe mitjançant la funció toInteger().
    - En cas que la mida del vector num sigui superior a 32, el transformo com a un vector de mida 64 amb extensió de signe mitjançant la funció toLong().
- Depenent de la nova mida del vector num, també transformo el valor de numPos utilitzant l'operació % perquè quedi un nombre mòdul amb la mida vàlid i pugui fer així el desplaçament correctament.
- Un cop tinc el vector transformat a la mida corresponent, creo un vector anomenat resultat amb la nova mida del vector num.
- Després faig una crida a la funció anterior amb els paràmetres num i numPos d'aquesta funció i ho guardo al vector resultat. Això em dóna com a resultat el vector amb les posicions desplaçades. Ara falta fer l'extensió de signe.
- Per a fer l'extensió de signe, he creat un if que compari si a la posició [num.length – numPos] del vector num hi ha un 1
    - En cas que així sigui, farem l'extensió de signe amb 1's mitjançant un for que va des de la posició [num.length – numPos] fins a la posició num.length
        - A cada iteració del for, es modifica el valor de la posició del vector resultat per 1.
    - En cas contrari, retornarem el vector ja que les posicions no modificades han quedat inicialitzades a 0.

2. Conclusió

Aquesta pràctica m'ha servit per a adquirir més coneixements amb el llenguatge de programació Java i aprendre sobre les diferències que té aquest amb els altres llenguatges.
Com ara a tractar cadenes de caràcters llegint els valors i passant-los a enters...
També he fet ús d'altres funcions implementades anteriorment per a reduir el tamany del codi i reutilitzar-lo.
He aprés a solucionar múltiples problemes de diferents maneres i a comprovar la forma més òptima de cadascuna d'elles.

3. Bibliografia

He consultat alguns dubtes en les següents pàgines:

- http://stackoverflow.com
- http://github.com

També he mirat els apunts teòrics de l'assignatura i he consultat algunes citacions del llibre (The Art and Science of Java, penjat al Campus Virtual).