Correzione della Teoria dell'esame di Programmazione del 14/7/2015

Nome e cognome: **Risposte su questo foglio**

1) Dire se il seguente programma è corretto o meno. Spiegare la risposta. Si consiglia di corredare le spiegazioni col disegno delle relazioni tra le variabili del programma durante l'esecuzione.

```
int** F(int* x){int**p=&x; (*p)++; return p;}

main(){int A[]={0,1,2,3,4},*q=A+2,*y=q+1; y=*F(y); *y=(*q)+1; cout<<*(A+2)<<*(A+3)<<*(A+4)<<endl;}
```

Esattamente come nell'analogo esercizio del 30/6, anche qui Frestituisce il valore di p che è l'indirizzo di x che è variabile locale di F. Il valore di p è quindi un dangling pointer e quindi F è sbagliata.

2) Si assuma che un programma venga sviluppato da 3 programmatori diversi, P1, P2 e P3 e che Pi produca la Parte_i del programma. Assumete che

i) la Parte_1 contenga la definizione di una funzione f1 che serva alle altre Parti,

ii)la Parte_2 contenga la definizione del tipo struct A{....} che serva alle altre Parti,

iii) la Parte_3 contenga la definizione della funzione f3 che serva anche alle altre parti.

Specificate:

a) quali file deve produrre ciascun programmatore?

Ogni programmatore deve produrre 2 file, un file Parte_i.cpp con le definizioni di tutte le funzioni che appartengono a questa parte ed un Parte_i.h che contiene il prototipo della funzione da esportare oppure la definizione completa della struttura A (se si tratta di P2).

b) come deve distribuirequesti file agli altri programmatori?

Ogni Pi deve dare il suo file Parte_i.h agli altri 2 programmatori

c) quali istruzioni di include devono essere presenti nei file?

all'inizio di ciascun file .cpp occorre avere

#include "Parte_1.h"

#include "Parte_2.h"

#include "Parte_3.h"

d) cosa potrà fare ciascun programmatore per fare il debugging della propria Parte?

deve compilare il proprio file .cpp con l'opzione -c che non linka e produce un file .o. In questo modo è possibile accorgersi dei propri errori anche in assenza delle altre parti.

3) Considerate il seguente frammento di programma che esegue una conversione tra puntatori:
int a=10, *pa=&a;
float * pb=reinterpret_cast<float*>(pa);
cout<<*pa<<' '<<*pb<<endl;
cosa produce la sua compilazione e, in caso di successo, la sua esecuzione? Motivate (brevemente) la risposta,

*il cast è perfettamente corretto e causa solamente il fatto che l'R-valore di pa venga assegnato a pb. Quindi pa e pb puntano alla stessa zona di memoria e, visto il loro tipo, pa la vedrà come contenente un intero (4 byte) mentre pb la vedrà come contenente un float (ancora 4 byte). Quindi stampare *pa comporta la stampa di 10 a video, mentre la stampa di *pb comporta la stampa di un valore decimale che è quello che risulta dall'interpretazione dei bit della rappresentazione interna dell' intero 10 come se fosse invece un floating point. In generale quindi, dopo il 10 verrà stampato un decimale che non ha nulla a che vedere con 10.*