(2a) Dato int T[3][5][6], dire che tipo ha l'espressione: *(T[8]-2)

(2b) Dato int T[3][5][6], dire che tipo ha l'espressione: ((*T)+8)[-2]

(1) Cercate di scrivere una PRE ed una POST sensate per il seguente programma ricorsivo:

```
int M(nodo *R, int *P, int dimP)
{
if(!dimP || !R)
  return 0;
int z=0;
if(R->info==*P)
{ z=1; P++; dimP--; }
int a=M(R->left, P, dimP);
int b=M(R->right, P, dimP);
if(a>=b)
  return a+z;
else
  return b+z;
}
```

(1) Data la seguente funzione ricorsiva, inserire appropriate PRE e POST

```
//PRE= ??
int F(Nodo* a) {
if(!a) return 0;
if (a->left) return 1+F(a->left);
if(a->left || a->right) return 2+F(a->left)+F(a->right);
return 3+F(a->left)+F(a->right);
} //POST=??
```

(2a) Considerare il seguente programma e dite se è corretto o no. Se pensate che sia corretto, spiegate i passi dell'esecuzione. Se pensate che sia sbagliato, spiegate con precisione perché.

```
int* f(int **p){int b=3,*x=&b; **p=*x; *x=**p; return x; }
main() {int y=5, b=2,*q=&b; *f(&q)=y*2; cout<<y<<b<<*q;}
```

(2b) Considerare il seguente programma e dite se è corretto o no. Se pensate che sia corretto, spiegate i passi dell'esecuzione. Se pensate che sia sbagliato, spiegate con precisione perché.

```
int* f(int **p){int b=3,*x=&b; **p=*x; x=*p; return x; }
main() {int y=5, b=2,*q=&b; *f(&q)=y*2; cout<<y<<b<<*q;}
```

3) Un programma può essere scritto su più file. Ogni file può venire compilato da solo, ma alla fine, tutti i file dovranno venire compilati assieme. Supponiamo di avere un programma scritto su due file e supponiamo che le funzioni scritte su ciascuno dei due file usino uno stesso tipo struttura P ed una stessa funzione f che sono come segue:

```
struct P{int a,b; P* next;};
P* f(P* x){.........}
```

Esattamente cosa di P e cosa di f deve essere scritto su ciascun file? Spiegare brevemente la risposta.

(1) Considerate il seguente programma. In caso pensiate sia corretto, spiegate cosa calcola e cosa stampa.

```
int** f(int * & p){int**x=&p; (*((*x)+1))++ ; p--; return x; }
main() {int b[]={2,3,4,5},*q=b+1; **f(q)=*q;
cout<<b[0]<<b[1]<<b[2]<<b[3];}
```

(2) Si dia PRE e POST alle seguenti 2 funzioni ricorsive:

```
int F(nodo*L)
{if(!L) return 0;
int x= G(L);
int y=F(L->next)
if(x>=y)
  return x;
else
  return y;
}
```

(1) Considerate il seguente programma. In caso pensiate sia corretto, spiegate cosa calcola e stampa.

```
int*& f(int * & p){int*& x=p; ++x; ++p; return x; }
main() {int b[]={2,3,4,5},*q=b; f(q)=b;
cout<<*q<<b[0]<<b[1]<<b[2]<<b[3];}
```

(2) Si trovino PRE e POST appropriate per la seguente funzione :

```
nodo* F(nodo*L)
{
if(!L->next) return L;
nodo*x=L->next;
L->next=x->next;
x->next=F(L);
return x;
}
```

(3) Si consideri la seguente dichiarazione: char X[3][4][4][5]
a) che tipo ha (*X)[-3] ?
b) (assumendo che X abbia valore X) che tipo e che valore ha *(X[-2])+1 ?

3) Data la seguente dichiarazione char X[10][5][10][5]; che tipo e che valore ha l'espressione seguente?
(X[3][-5]) – 4
Si assuma che l'R-valore di X sia X.

(1) Scrivere PRE e POST opportune per la seguente funzione ricorsiva:

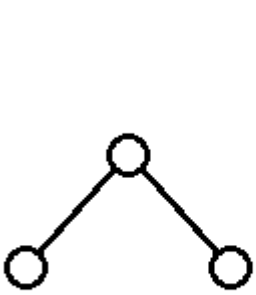
```
//PRE= ??
bool H(nodo*L, int & k)
{ if(!L) { k=1; return false; }
if(H(L->next, k)) {k=k+1; return false;}
else return true;
}
POST= ??
```

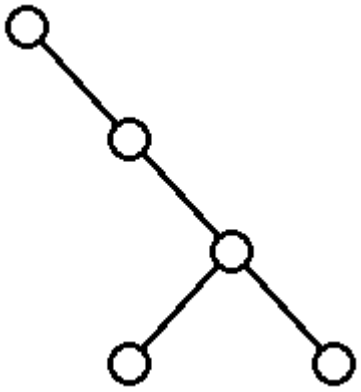
(1) Data la dichiarazione char X[4][8][10], qual è il tipo di (*X)[-2] e quale dimensione ha l'oggetto puntato da questo puntatore? Inoltre che valore ha l'espressione (*X)[-2] rispetto al valore di X?

(
(1) Data la seguente funzione ricorsiva:

```
inf F(nodo* a) {
if(!a) return false;
if(a->left && a->right) return F(a->right) || F(a->left);
return !( F(a->left) || F(a->right));
}
```

Dire cosa calcola F invocate sulle radici dei due alberi raffigurati. Sulla base di questi valori, specificare un'appropriata coppia di PRE e POST per la funzione F.





(1) Dato un array int X[3][4][5][10], si chiede di:
i) specificare il tipo di X e la dimensione dell'oggetto puntato
ii) specificare il valore e il tipo di (*X)+2 e di *(X+2)
iii) specificare il valore e il tipo di *(X[-8])+8

Considerare il seguente programma:

```
int* f(int **p) {
    int b = 3, *x = &b;
    **p = *x;
    *x = **p;
    return x;
}

int main() {
    int y = 5, b = 2, *q = &b;
    *f(&q) = y * 2;
    cout << y << b << *q;
}
```

a) Il programma è corretto?
b) Se sì, spiegare i passi dell'esecuzione.
c) Se no, spiegare con precisione perché.

Dato il seguente programma:

```
int** f(int *& p) {
    int **x = &p;
    (*((*x)+1))++;
    p--;
    return x;
}

int main() {
    int b[] = {2,3,4,5}, *q = b+1;
    **f(q) = *q;
    cout << b[0] << b[1] << b[2] << b[3];
}
```

a) Il programma è corretto? b) Se sì, cosa calcola e cosa stampa? c) Se no, spiegare gli errori.

Dato il codice:

```
#include <stdio.h>

void reverse_array(int **arr, int size) {
    // IMPLEMENTA LA FUNZIONE QUI
}

int main() {
    int size = 5;
    int *arr = malloc(size * sizeof(int));
    for (int i = 0; i < size; i++) {
        arr[i] = i + 1;
    }

    reverse_array(&arr, size);

    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }

    free(arr);
    return 0;
}
```

Implementa la funzione reverse_array che inverte l'ordine degli elementi nell'array utilizzando solo manipolazione di puntatori.

Dato il codice:

```
// PRE:
// POST:
void compress_string(char* s, char* result) {
    // IMPLEMENTA LA FUNZIONE QUI
}

int main() {
    char s[100] = "aabbbccccaaa";
    char result[100];
    compress_string(s, result);
    printf("%s\n", result);
    return 0;
```

```
}
```

Implementa la funzione compress_string che comprime ricorsivamente una stringa sostituendo le sequenze di caratteri ripetuti con il carattere seguito dal numero di ripetizioni. Ad esempio, "aabbbccccaaa" diventa "a2b3c4a3". Fornisci PRE e POST condizioni appropriate.

Dato il codice:

```
void mystery(int *arr, int size) {
    for (int i = 0; i < size / 2; i++) {
        int temp = arr[i];
        arr[i] = arr[size - 1 - i];
        arr[size - 1 - i] = temp;
    }
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);
    mystery(arr, size);
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
}
```

a) Cosa fa la funzione mystery? b) Il codice è corretto? c) Cosa stamperà il programma?

Dato il codice:

```
void reverse(char *str) {
    if (*str) {
        reverse(str + 1);
        putchar(*str);
    }
}

int main() {
    char s[] = "Hello";
    reverse(s);
}
```