esercizi su funzioni

- passaggio dei parametri
- restituzione dei risultati
- invocazione

```
int * f(int & x){x=5; return &x;}
main() { int y=1;
*f(y)=25;
cout<<y<<endl;
}
```

è corretto e stampa 25

x è un alias di *y* ed *f* ritorna un puntatore a *y*

quindi *y* diventa 5 in *f* e poi 25

```
int & f(int & x){x=5; return &x;}

main()
{
int y=1;
*f(y)=25;
cout<<y<<endl;
}
```

ERROR: In function ‘int main()’: invalid type argument of ‘unary *’

```
int **f(int * x){*x=5; return &x;}
main() { int y=1;
**f(&y)=25;
cout<<y<<endl;
}
```

restituisce un dangling pointer, ma nessun warning

```
int * f(int * & x){*x=5; return x;}
main()
{ int y=1, x;
x=*f(&y);
cout<<y<<‘ ‘<<x<<endl;}
```

In function ‘int main()’: inizialization of non-const reference ‘int *&’ from r-value ‘int *’ in passing argument 1 of ‘f(int * &)’ .

```
int & f(int * & x){*x=5; return *x;}
main() {

int y=1, x; int *z=&y;
x= f(z);
cout<<y<<‘ ‘<<x<<endl;

}
```

CORRETTA: stampa 5 5

f ritorna un alias di *y* e x assume il valore di y

```
int & f(int * & x){*x=5; return *x;}
main() { int y=1, *z=&y;
f(z)=25;
cout<<y<<endl;
}
```

CORRETTA: stampa 25, infatti *f* ritorna un alias di *y*, quindi *y* riceve 25

attenzione:

....f(int &x)

int a=1, *p=&a;
....f(*p)..... // ok

esercizio 1

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

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

esercizio 2: ordine di valutazione !

```
int *& F(int** & p){int*x=(*p)+2; *p=x+1; return *p;}

main()
{
int X[5]={0,1,2,3,4}, *q=X+1, **p=&q;
F(p)=q-2;
cout<<*q<<**p<<endl;
}
```

esercizio 3

```
int k=5, *z=&k;

*f(&z)=k+5;

cout<<*z <<endl;
```

vogliamo una f() t. c. stampi 10