

Nome:

Cognome:

Matricola:

## Domanda 1 – Punti 2

Nella compilazione separata di un progetto composto da più sorgenti:

Scegli una o più alternative:

- ☒ a. Quando si dà il comando make senza parametri vengono ricompilati tutti i file modificati rispetto all'ultima generazione di file oggetto
- ☐ b. Viene generato un solo file oggetto
- ☒ c. Vengono invocati in ordine il preprocessore, il compilatore e il linker
- ☐ d. È obbligatorio inserire un target clean all'interno del makefile

## Domanda 2 – Punti 2

Indicare la o le risposte corrette

Scegli una o più alternative:

- ☒ a. I file .h forniscono un'interfaccia del modulo
- ☒ b. Un file nome.h può essere inserito in diversi file .cc
- ☒ c. Separare il codice in più moduli rende più manutenibile il progetto
- ☐ d. I file .h vanno inseriti solo se necessari per la compilazione dei file .cc, altrimenti viene generato un warning di tipo W\_H\_USELESS

## Domanda 3 – Punti 2

Supponendo di voler documentare il proprio progetto tramite doxygen

Scegli una o più alternative:

- ☐ a. Per inserire una descrizione breve si usa il comando @SHORT\_DESC, per inserire quella estesa @LONG\_DESC
- ☒ b. Con il comando @param si possono specificare i parametri di una funzione
- ☐ c. Se si sceglie di usare doxygen in un file, è necessario commentare tutte le funzioni (ma non le variabili)
- ☒ d. È possibile generare documentazione in più formati (PDF, LaTeX ecc)

Nome:

Cognome:

Matricola:

## Domanda 4 – Punti 3

Dire cosa stampa il seguente codice

```
#include <iostream>

using namespace std;

int f2(int n) {
    cout << n << " ";
    return --n; }

int f3(int n) {
    cout << n << " ";
    if (n % 2) {
        return n-1;
    }
    return f2(n-1); }

int f(int n) {
    if (n == 0) {
        return 0;
    } else if (n < 4) {
        cout << n << " ";
        f(f3(n--));
    } else if (n < 2) {
        return f2(n-1);
    } else {
        return f(n-1);
    }
    return n; }

int main() {
    cout << f(5);
}
```

Risposta: 3 3 2 2 1 2

Nome:

Cognome:

Matricola:

## Domanda 5 – Punti 3

Dire cosa stampa il seguente codice

```
#include <iostream>

using namespace std ;

int f1(int p) {
    return (p*p); }

int f2(int p) {
    return (p*2); }

void f(int (*fun1)(int), int (*fun2)(int), int* v, int n) {
    for(int i=0;i<n;i++)
        if(i<2)
            v[i]=fun1(v[i]+1);
        else
            v[i]=fun2(v[i]+v[i-2]);
}

int main() {
    int a[]={0,1,1,0};
    f(f1,f2,a,4);
    for(int i=0;i<4;i++) {
        cout<<a[i]<<" ";
    }
    cout<<endl;
}
```

Risposta: 1 4 4 8

Nome:

Cognome:

Matricola:

## Domanda 6 – Punti 3

Supponendo di avere il seguente codice

```
#include <iostream>
```

```
using namespace std;
```

```
int f1(int n) {  
    XXX  
}
```

```
int f2(int n) {  
    YYY  
  
}
```

```
int main() {  
    int a[] = {1, 4, 2, 3, 9};  
  
    for (int i = 0; i < 4; i++) {  
        if (i % 2) {  
            a[i] = f1(i);  
        } else {  
            a[i] = f2(a[i]);  
        }  
        a[i] = a[i] | a[i+1]; }  
  
    for (int i=0; i < 5; i++) {  
        cout << a[i] << " ";  
    }  
}
```

Scrivere le due righe di codice mancanti al posto di XXX e YYY (una riga per XXX, una per YYY), affinché il programma stampi

4 2 3 15 9

Risposta:    return n\*2, return n/2

Nome:

Cognome:

Matricola:

## Domanda 7 – Punti 5

Dato il tipo di dato lista semplice definito sotto e le sue primitive, scrivere la funzione `lista_somma_elemento(lista,int)` che prende in ingresso una lista e un intero `p`, e somma il numero che si trova alla posizione `p` a tutti gli elementi della lista, restituendola come risultato:

```
struct elem{  
    tipo_inf inf ;  
    elem* pun ;  
};
```

```
typedef elem* lista ;
```

Esempio:

Lista -> 5 2 4 3

Chiamata -> `somma_elemento(lista,2)`

Risultato -> 9 6 8 6 (ovvero la somma dell'elemento in posizione 2 (che è 4) con tutti gli altri elementi)

Nome:

Cognome:

Matricola:

## Domanda 8 – Punti 6

Data la seguente definizione di lista doppia intera, scrivere la funzione booleana `palindroma(lista l)` che restituisce `true` se `l` è palindroma ovvero le sequenze dei valori della lista lette dal primo all'ultimo e dall'ultimo al primo sono le stesse, false altrimenti (ad esempio `[1,4,2,4,1]` è palindroma, `[1,4,2,3,1]` non è palindroma)

```
struct elem
```

```
{
```

```
    int inf;
```

```
    elem* pun ;
```

```
    elem* prev;
```

```
};
```

```
typedef elem* lista ;
```

Nome:

Cognome:

Matricola:

## Domanda 9 – Punti 6

Data la seguente definizione di grafo:

```
struct adj_node { int node;  
    int weight; struct  
    adj_node* next;  
};  
  
typedef adj_node* adj_list;  
  
typedef struct {  
    adj_list* nodes;  
    int dim;  
} graph;
```

Scrivere la funzione `bool even_path(graph g, int x, int y)` che restituisce `true` se esiste un cammino da `x` a `y` nel grafo che attraversa solo nodi con id pari (esclusi `x` e `y`), `false` altrimenti. La funzione può far uso delle primitive per la gestione del grafo.