

NOME:

COGNOME:

MATRICOLA:

[1] Punti 2 – Domanda a risposte multiple (sono possibili 0 o più risposte)Indicare quali sono gli effetti della direttiva `#include<nome_file>`:

- a. Viene indicato al linker che il file `nome_file.o` dovrà essere collegato dinamicamente al file oggetto prodotto dalla compilazione del file sorgente in cui appare la direttiva
- b. E' una direttiva al preprocessore
- c. Il contenuto di `nome_file` viene inserito interamente nel file sorgente a partire dal punto in cui compare la direttiva
- d. Nessuna delle precedenti

[2] Punti 2 – Domanda a risposte multiple (sono possibili 0 o più risposte)Indicare quali delle seguenti affermazioni è vera in caso di invocazione del comando `make` in presenza di un Makefile contenente la seguente regola:

```
target.o: prerequisito1 prerequisito2
    comando1
    comando2
```

- a. Le regole implicite di compilazione permettono di evitare di inserire tra i comandi il comando di compilazione
- b. Se il target è più recente del file di nome `prerequisito2`, i comandi devono essere rieseguiti
- c. Se `prerequisito1` rappresenta a sua volta un target nel Makefile, le sue regole devono essere controllate
- d. Le regole implicite di compilazione permettono di evitare di inserire tra i prerequisiti l'eventuale file sorgente di nome `target.c`

[3] Punti 2 – Domanda a risposte multiple (sono possibili 0 o più risposte)

In un programma sviluppato su più file/moduli, indicare cosa deve essere incluso nella parte di header di ogni modulo:

- a. Le definizioni delle funzioni da esportare per permettere l'accesso ai servizi del modulo
- b. Le strutture dati che servono per l'implementazione interna del modulo
- c. Le dichiarazioni delle funzioni esportare per permettere l'accesso ai servizi del modulo
- d. Le definizioni delle funzioni che servono per l'implementazione interna del modulo

[4] Punti 3 – Domanda a risposta aperta

Cosa stampa il seguente programma: _____

```
#include <iostream>
using namespace std;
void stampa(int n) {
    if (n==0) return;
    if (n % 2)
        stampa(n-1);
    else{
        stampa(n-2);
        cout << n <<" ";
        return; } }
```

NOME:

COGNOME:

MATRICOLA:

```
int main() {  
    int x = 9;  
  
    stampa(x); }
```

[5] **Punti 3** – Domanda a risposta aperta

Cosa stampa il seguente programma: _____

```
#include <iostream>  
using namespace std ;  
  
void f(int *p, int *&rp) {  
    int *tmp = p;  
    p = rp;  
    *tmp = 20;  
    *p = 10;  
    rp = p;  
}  
  
int main() {  
    int a = 30, b = 15;  
    int *punt = &a;  
    int *prp = &b;  
  
    f(punt, prp);  
    cout << a << " " << *punt << " " << b << " " << *prp << " "  
    << endl;  
  
}
```

[6] **Punti 3** – Domanda a risposta aperta

Si assuma di compilare i seguenti file, fun.cc e main.cc, in un unico progetto.

La compilazione va a buon fine? _____

Se no, quale errore viene segnalato? _____

Se si, che cosa stampa l'eseguibile che si ottiene? _____

fun.cc:

```
#include <iostream>  
using namespace std ;  
  
extern int a;  
  
void fun()  
{  
    cout << "La variabile a vale: " << a << endl ;  
}
```

main.cc

```
#include <iostream>  
using namespace std ;
```

NOME:

COGNOME:

MATRICOLA:

```
int a = 2;
void fun() ;
int fun(int);

int main()
{
    int a = 3;
    a = fun(a);
    fun() ;
    return 0 ;
}

int fun(int x){
    return x+a;
}
```

[7] Punti 4 – Scrittura di codice

Date le seguenti dichiarazioni per un binary search tree con chiave intera, scrivere il codice della primitiva per la ricerca di un nodo **bst_search**, assumendo che i nodi siano ordinati in *ordine decrescente* (dal più grande al più piccolo):

```
typedef int tipo_key;

struct bnode {
    int key;
    bnode* left;
    bnode* right;
    bnode* parent;
};

bnode* bst_search(bnode* b, int k){

}
```

[8] Punti 4 – Scrittura di codice

NOME:

COGNOME:

MATRICOLA:

Date la seguente dichiarazione di lista e primitive, scrivere la *procedura ricorsiva* `stampa_ric` che stampa il contenuto della lista su un'unica riga (valore degli elementi separati da uno spazio) e al termine un newline:

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

typedef elem* lista ;

int head(lista p); //restituisce il contenuto della testa
lista tail(lista p); //restituisce la coda della lista

void stampa_ric(lista p)
{

}

}
```

[9] **Punti 4** – Scrittura di codice

Date la seguente dichiarazione di coda, completare la primitiva `enqueue` che aggiunge un elemento alla coda.

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

typedef elem* lista;

typedef struct{
    lista head;
    elem* tail;} coda;

coda enqueue(coda c, tipo_inf i){
```

NOME:

COGNOME:

MATRICOLA:

}

[10] Punti 4 – Scrittura di codice

Date le seguenti dichiarazioni di albero e relative primitive, scrivere la funzione booleana `path(node*e, int x)` che restituisce true se esiste un cammino dal nodo e un nodo con valore x, false altrimenti.

```
struct node {  
    tipo_inf inf;  
    node* firstChild;  
    node* nextSibling;  
};
```

```
typedef node* tree;
```

```
tipo_inf get_info(node*); //restituisce il contenuto  
informativo del nodo
```

```
node* get_firstChild(node*); //restituisce il primo figlio del  
nodo
```

```
node* get_nextSibling(node*); //restituisce il prossimo  
fratello del nodo
```

```
bool path(node* n, tipo_inf v){
```

}