# Informatica A

**Cognome (IN STAMPATELLO) Matricola**

**Nome Data**

Istruzioni

1. Non separate questi fogli. Scrivete la soluzione **solo** **sui fogli distribuiti**, utilizzando il retro delle pagine in caso di necessità.
2. **È possibile scrivere a matita** (e non ricalcare al momento della consegna).
3. Ogni parte non cancellata sarà considerata parte integrante della soluzione.
4. È **vietato** utilizzare **calcolatrici** o **telefoni**. Chi tenti di farlo vedrà **annullata** la sua prova.
5. Non è ammesso consultare **libri** o **appunti** o altro.
6. Qualsiasi **tentativo** di comunicare con altri studenti comporta **l’espulsione** dall’aula.
7. È possibile **ritirarsi senza penalità** lasciando il tema d’esame con nome e cognome.
8. Non è possibile lasciare l’aula conservando il tema della prova in corso.
9. Tempo a disposizione: 2 ore e 30 minuti

### Valoredegli esercizi, voti parziali e voto finale:

**Esercizio 1 ( 2 punti )**

**Esercizio 2 ( 4 punti )**

**Esercizio 3 ( 4 punti )**

**Esercizio 4 ( 8 punti )**

**Esercizio 5 (10 punti )**

**Esercizio 6 ( 4 punti )**

**Totale: ( 32 punti )**

## Esercizio 1 (2 punti)

1. Si costruisca la tabella di verità della seguente espressione booleana, badando alla precedenza tra gli operatori logici.

**( A or B ) or ( not A or ( (not B) and C ) )**

1. Si stabilisca il minimo numero di bit sufficiente a rappresentare in complemento a due entrambi i numeri A = 121dec e B = –95dec

Si converta A e B in CP2 usando bit

Si calcoli quindi (A+B) e (A–B) in complemento a due e si indichi se si genera riporto sulla colonna dei bit più significativi e se si verifica overflow.

## Esercizio 2 ( 4 punti )

Si consideri il seguente schema, relativo a un sistema di gestione di una compagnia di affitto di camere:

CAMERA (Codice, Modello, MetriQuadri, Categoria, PrezzoGiorno)

CLIENTE ( CF, Nome, Cognome, Indirizzo, Citta)

AFFITTO (CodiceCamera, CF, DataInizio, DataFine)

Dove l’attributo Categoria nella tabella CAMERA è un valore numerico che va da 1 a 5 che attesta la qualità della camera (1 è la più economica mentre 5 è la più costosa).

Esprimere la query che trova i clienti che hanno affittato una e una sola camera di categoria maggiore o uguale a quattro.

Si consideri il seguente schema, relativo a un sistema di gestione di una compagnia di affitto di camere:

CAMERA (Codice, Modello, MetriQuadri, Categoria, PrezzoGiorno)

CLIENTE ( CF, Nome, Cognome, Indirizzo, Citta)

AFFITTO (CodiceCamera, CF, DataInizio, DataFine)

Esprimere la query che trova i clienti che non hanno mai affittato due volte la stessa camera.

**Esercizio 3** ( 4 punti )

Si dica cosa stampa il seguente programma, completando la definizione di **MATR** con la propria matricola.

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#define MATR "......"

void f(char \* str) {

if (str == NULL)

return;

else {

printf("%c", str[0]);

str=str+1;

if(str == '\0')

return;

f(str-1);

printf("%c", str[0]);

return;

}

}

int main() {

char m[7] = MATR;

f(m+4);

return 0;

}

## Esercizio 4 ( 8 punti )

Scrivere una funzione

int CercaUniSottomatrici(int m[N][M], int k, int h)

che restituisce 1 se all’interno della matrice N x M data in input (assumendo N <= M) esistono una sottomatrice quadrata di dimensione k x k ed un’altra di dimensione h x h (assumendo k < h con k > 1) contenenti lo stesso numero di valori uguali ad 1. Altrimenti la funzione restituisce 0.

Si consiglia di utilizzare funzioni ausiliarie.

Esempi:

1 2 0 1 0

3 0 2 0 3 ---> La funzione restituisce 0 per k = 2 ed h = 3

1 2 1 1 2

1 2 0 2 0

3 0 2 0 3 ---> La funzione restituisce 1 per k = 2 ed h = 3

1 2 1 1 2

## Esercizio 5 ( 10 punti )

Si consideri la seguente definizione di lista:

**typedef struct EL {**

**int dato;**

**struct EL \* next;**

**} nodo;**

**typedef nodo \* lista;**

Data una lista x1  x2 … xn, si definisce *folding* una operazione che modifica la lista in modo tale da riordinare i suoi elementi come segue: x1 xn x2 xn-1 x3 …

Ad esempio:

In: 1 2 3 4 5 6 NULL

Out*:* 1 6 2 5 3 4 NULL

Scrivere una funzione

**lista fold(lista lis)**

che, presa in input una lista, restituisce una lista ottenuta applicando l’operazione di *folding* sulla lista data in input.

Si scriva quindi una porzione dei **main.c** contenente l’invocazione della funzione **fold** e la sola dichiarazione di tutte le variabili coinvolte.

Si codifichi una funzione che **cancellaTriplette** che rimuove da una lista tutte le sequenze di almeno tre elementi dispari

Ad esempio

Input: **1 -> 40 -> 15 -> 16 -> 3 -> 15 -> 5 -> 6 -> NULL**

La funzione restituisce

**1 -> 40 -> 15 -> 16 -> 6 -> NULL**

Esercizio 6 **( 4 punti )**

Si consideri la seguente di albero binario:

**typedef struct nodeS { int v;**

**struct nodeS \* left, right; } node;**

**typedef node \* tree;**

Si scriva una funzione “simmetrici” che dati due alberi binari restituisce 1 se i due alberi sono simmetrici, 0 altrimenti.

int simmetrici(Tree t1, Tree t2)

Esempio:

1

|

|-------------|

3 2

|----------|

5 4

È simmetrico con

1

|---------------|

2 3

|-------|

4 5