Informatica A

**Cognome Matricola**

**Nome Firma**

Valore indicativo degli esercizi, voti parziali e voto finale:

**1 ( 2 p. )**

**2 ( 4 p. )**

**3 ( 4 p. )**

**4 ( 4 p. )**

**( 14 p. )**

1 (2 p.)

Build the truth table of the following Boolean expression.

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

Represent in two's complement the binary numbers equivalent to A = 77dec and B = –123dec, then compute the sum (A+B) and the difference (A–B) in two's complement.

Esercizio 2 (4 punti)

SCRIVI the printed output of the following code.

#include<stdio.h>

#define N 8

void t(int n, int p, int\*i, int\*f) {

if( i>=f )

printf(" NOT FOUND \n");

else if( n == \*i )

printf(" : odd[%d] \n",p);

else

t( n, p+1, i+1, f );

}

void d( int n, int \*v ) {

if( n%2 )

t( n, 0, v, v+N );

else {

printf(" -> %d", n/2);

d(n/2, v);

}

}

int main(){

int odd[4\*N]={1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47},

x[N]={7,18,24,17,88}, i=0;

while( x[i] > 0 ) {

printf("[%d]", x[i]);

d( x[i++], odd );

}

return 0;

}

Esercizio 3 (4 punti)

Consider the following type declarations describing the data structure needed for the management of a dentist’s schedule.

typedef struct {

int day, month, year;

} date;

typedef struct {

int hour, minute;

} time;

typedef struct {

char SSN[100], surname[100], name[100];

data BirthDate;

} person;

typedef struct {

person patient[1000];

int patientNumber; //number of array cells containing valid data

} patientList;

typedef struct {

char PatientSSN [100];

date d;

time t;

} visit

typedef struct {

visit visits[1000];

int visitsNumber; //number of array cells containing valid data

} VisitList;

Write a C function

patientList f(patientList P, visitList V);

that returns the list of patients that had a visit to the dentist on their birthday.

Esercizio 4 **(4 punti)**

A matrix is called a “chessboard” if it contains zeros and non-zero values disposed as dark and light squares in a chessboard.

Consider the following matrixes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 5 | 0 | 21 | 0 | 7 |
| 6 | 0 | 4 | 0 | 2 | 0 |
| 0 | 4 | 0 | 11 | 0 | 8 |
| 55 | 0 | 33 | 0 | 55 | 0 |
| 0 | 5 | 0 | 21 | 0 | 9 |
| 22 | 0 | 1 | 0 | 32 | 0 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 22 | 0 | 1 | 0 | 32 | 0 |
| 0 | 5 | 0 | 21 | 0 | 7 |
| 6 | 0 | 4 | 0 | 2 | 0 |
| 0 | 4 | 0 | 11 | 0 | 8 |
| 55 | 0 | 33 | 0 | 55 | 0 |
| 0 | 5 | 0 | 21 | 0 | 9 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 5 | 0 | 21 | 0 | 7 |
| 0 | 4 | 0 | 2 | 0 | 4 |
| 0 | 4 | 0 | 11 | 0 | 8 |
| 55 | 0 | 33 | 0 | 55 | 0 |
| 0 | 5 | 0 | 21 | 0 | 9 |
| 22 | 0 | 1 | 0 | 32 | 0 |

The first one and the secondo one are “chessboard” matrixes, the third one is not a “chessboard”.

Write a function

int f(int M[][N])

that receive as an input a NxN square matrix and returns 1 if the matrix is a “chessboard”, 0 otherwise.