```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <bitset> // Pour l'affichage binaire.
using namespace std;

int signe(signed char c)
{
    return c>>7;
}

void q1()
{
    signed char val1 = bitset<8>("11010101").to_ulong();
    signed char val2 = bitset<8>("10110111").to_ulong();
    signed char resultat = val1 + val2;
    bool debordement = signe(val1) == signe(val2) && signe(val1) != signe(resultat);
    cout << "1a) " << bitset<8>(resultat) << "  debordement: " << (debordement?"oui":"non") << endl;
    // 10001100

    float x;
    int& x_i = *(int*)&x;
    x = 2.5f;
    cout << "1b) "
        << bitset<1>(x_i>>31) << "  "
        << bitset<8>(x_i>>23) << "  "
        << bitset<23>(x_i)
        << endl;
    //0  10000000  01000000000000000000000

}

int f(int x, int& y)
{
    int t = x + y;
    x = x + 4;
    y = y + 8;
    return t;
}

void q2a()
{
    int a = 1;
    int b = 2;
    int c = f(a, b);
    cout << a << ' ' << b << ' ' << c;
    //1 10 3
}

struct S {
    int a, b;
};

void q2b()
{
    S a;
    char* b = new char[10];
    cout << sizeof(a) << ' ' << sizeof(b) << ' ' << sizeof(b[0]);
    //8 4 1

}

void q2()
{
    cout << "q2a) "; q2a(); cout << endl;
    cout << "q2b) "; q2b(); cout << endl;
}

// q3
```

```
// a) F b)F c) F d) V

// Programme pour les dates de fête...
//#define DYNAMIQUE // Pour les questions 3 e) et suivantes.
//#define BINAIRE // Pour la quesiton 3 g)
static const int MAX_PERSONNES = 10;
struct Date {
    int annee, mois, jour;
};
struct Personne {
#ifndef BINAIRE
    string nom;
#else
    char nom[40];
#endif
    Date naissance;
};
#ifndef DYNAMIQUE
struct ListePersonnes {
    Personne personnes[MAX_PERSONNES];
    int nombreDePersonnes;
};
#else
struct ListePersonnes { // e)
    Personne* personnes;
    int nombreDePersonnes;
    int capacite;
};
#endif

void afficherPersonne(const Personne& personne)
{
    cout << personne.nom << ' '
        << personne.naissance.annee << '-'
        << personne.naissance.mois << '-'
        << personne.naissance.jour
        << endl;
}
// a)  1.5 points
bool estFete(const Date& naissance, const Date& aujourdhui)
{
    return naissance.mois == aujourdhui.mois && naissance.jour == aujourdhui.jour;
}
// b) 1.5 points
void afficherFetes(const ListePersonnes& listePersonnes, const Date& aujourdhui)
{
    for (int i = 0; i < listePersonnes.nombreDePersonnes; i++)
        if (estFete(listePersonnes.personnes[i].naissance, aujourdhui))
            cout << listePersonnes.personnes[i].nom << endl;
}
// c) 2.0 points
void ajouterPersonne(ListePersonnes& listePersonnes, const Personne& personne)
{
    listePersonnes.personnes[listePersonnes.nombreDePersonnes] = personne;
    listePersonnes.nombreDePersonnes++;
}
void q3d(ListePersonnes& listePersonnes)
{
    Personne personnes[] = {
        { "Un nom",        { 1920, 2, 12 } },
        { "Un autre",      { 1930, 3, 22 } },
        { "Encore",        { 1940, 4, 25 } },
        { "Puis un autre", { 1950, 5,  8 } }
    };
    for (int i=0; i<4; i++)
        ajouterPersonne(listePersonnes, personnes[i]);

    //{ d) 1.5 points
    Personne alanTuring = { "Alan Turing", { 1912, 6, 23 } };
```

```
    ajouterPersonne(listePersonnes, alanTuring);
    //}

    cout << listePersonnes.nombreDePersonnes << ' ';
    afficherPersonne(listePersonnes.personnes[listePersonnes.nombreDePersonnes - 1]);
}
#ifndef DYNAMIQUE
void q3()
{
    ListePersonnes listePersonnes = {};
    q3d(listePersonnes);
}
#else
// f) 3.points
void augmenterCapacite(ListePersonnes& listePersonnes, int nouvelleCapacite)
{
    Personne* temp = new Personne[nouvelleCapacite];
    for (int i = 0; i < listePersonnes.nombreDePersonnes; i++)
        temp[i] = listePersonnes.personnes[i];
    delete[] listePersonnes.personnes;
    listePersonnes.personnes = temp;
    listePersonnes.capacite = nouvelleCapacite;
}


void q3()
{
    ListePersonnes listePersonnes = {};
    augmenterCapacite(listePersonnes, 10);
    q3d(listePersonnes);


    #if BINAIRE
    {
        ofstream fichier("personnes.bin", ios::binary);
        fichier.write((char*)listePersonnes.personnes, listePersonnes.nombreDePersonnes * sizeof(Personne));
    }

    {
        // g) 2.5 points
        //  struct Personne {
        //       char nom[40];
        //       Date naissance;
        //   };

        Personne personne;
        ifstream fichier("personnes.bin", ios::binary);
        fichier.seekg(4 * sizeof(personne));
        fichier.read((char*)&personne, sizeof(personne));

        afficherPersonne(personne);
    }
    #endif
}
#endif

int main()
{
    q1();
    q2();

    q3();
}
```