Concours Informatique Luxembourgeois 2012
Epreuve Finale (05/07/2012)
Solutions

Tâche I - Marathon 50 points

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstdlib>
#include <algorithm>
#include <iomanip>
#include <sstream> // pour la conversion int to string

using namespace std;

const int MAX = 20000;
int c[MAX][MAX];

struct Coureur {
	int dossard;
	int age;
	int temps;
};


int longueur_lcs(Coureur x[],int m,Coureur y[],int n)
{
	for(int i = 0; i <= m; i++) c[i][0] = 0;
	for(int i = 0; i <= n; i++) c[0][i] = 0;
	for (int i = 1; i <= m; i++)
		for (int j = 1; j <= n; j++)
			if (x[i].age == y[j].age)
				c[i][j] = 1+ c[i-1][j-1];
			else
				c[i][j] = max(c[i][j-1], c[i-1][j]);
	return c[m][n];
}

string backtrack(Coureur x[],int m,Coureur y[],int n, int i, int j)
{
	if ((i==0) || (j==0))
		return "";
	else
		if (x[i].age == y[j].age)
		{
			ostringstream osstream1,osstream2;
			osstream1 << x[i].age;
			string sta = osstream1.str();
			osstream2 << x[i].dossard;
			string stdo = osstream2.str();
			string test = backtrack(x,m,y,n,i-1,j-1);
```

```
                  //cout << test + stdo + "-" + sta+ " " << flush;
                  return test + stdo + "-" + sta+ " ";
            }
            else
                  if (c[i][j-1] > c[i-1][j]) {
                        string test =backtrack(x,m,y,n,i,j-1);
                        //cout << test<< flush;
                        return test;
                  }
                  else{
                        string test =backtrack(x,m,y,n,i-1,j);
                        //cout << test<< flush;
                        return test;
                  }
}

struct compareCoureurParAge
{
   bool operator() (const Coureur & lhs, const Coureur & rhs) { return lhs.age > rhs.age; }
};

bool egaliteAge (Coureur i, Coureur j) {
  return (i.age==j.age);
}

bool found(Coureur x[],int m, Coureur c) {
      bool trouve;
      int i;

      trouve = false;
      i = 0;
      while (i <= m && ! trouve) {
            if (x[i].age == c.age) trouve = true;
            ++i;
      }
      return trouve;
}

void tassement(Coureur x[],int m,int &nouvelM) {
      int last;

      last = 0;
      for (int i = 1; i <= m; ++i)
      {
            if (!found(x,last, x[i])) {
                  last++;
                  x[last] = x[i];
            }
      }
      nouvelM = last;
}

int main(int argc, char **argv) {
      Coureur x[MAX+1], y[MAX+1];

    string nf, temps;
    cout << "Nom du fichier : ";
    //getline(cin,nf);
    cin >> nf;
    ifstream f(nf.c_str());
    if (f.is_open())
    {
            int dossard, i, heures, minutes, secondes, nbreCoureurs,
nouveauNbreCoureurs;
            char tmp;

            // lire le fichier d'entrée dans x et l'afficher
```

```
            f >> dossard;
            i = 0;
            while (dossard != -1)
            {
                  x[i].dossard = dossard;
                  f >> x[i].age;
                  f >> heures;
                  f >> tmp;
                  f >> minutes;
                  f >> tmp;
                  f >> secondes;
                  x[i].temps = heures*3600+60*minutes+secondes;
                  i++;
                  f >> dossard;
            }
            nbreCoureurs = --i;
            cout << "x\n";
//          for (int i = 0; i < nbreCoureurs; i++)
//                cout << setiosflags(ios::right) << setw(4) << i << setw(7) <<
x[i].dossard << setw(6) << x[i].age << setw(8) << x[i].temps << endl;

            // copier x dans y
            for (int i = 0; i < nbreCoureurs; i++) y[i] = x[i];

            // trier y sur les âges décroissants et l'afficher
            sort(y,y+nbreCoureurs,compareCoureurParAge());
            cout << "y\n";
//          for (int i = 0; i < nbreCoureurs; i++)
//                cout << setiosflags(ios::right) << setw(4) << i << setw(7) <<
y[i].dossard << setw(6) << y[i].age << setw(8) << y[i].temps << endl;

            // rendre les âges uniques
            tassement(y, nbreCoureurs, nouveauNbreCoureurs);
            cout << "y tasse\n";
//          for (int i = 0; i < nouveauNbreCoureurs; i++)
//                cout << setiosflags(ios::right) << setw(4) << i << setw(7) <<
y[i].dossard << setw(6) << y[i].age << setw(8) << y[i].temps << endl;


      cout << longueur_lcs(x,nbreCoureurs,y,nouveauNbreCoureurs) << endl;
//          cout << longueur_lcs(y,nouveauNbreCoureurs,x,nbreCoureurs) << endl;
            cout <<
backtrack(x,nbreCoureurs,y,nouveauNbreCoureurs,nbreCoureurs,nouveauNbreCoureurs)
<< endl;
//          cout <<
backtrack(y,nouveauNbreCoureurs,x,nbreCoureurs,nouveauNbreCoureurs,nbreCoureurs)
<< endl;
/*          diff(x,m,y,n,m,n);
            cout << endl;
            diff1(x,m,y,n,m,n);
            cout << endl;
            cout << diff2(x,m,y,n,m,n) << endl;
*/        f.close();
    }
    else cout << "Unable to open file" << endl;

    return EXIT_SUCCESS;
}
```

Tâche II - Spacer 50 points

```
program Project;
{$APPTYPE CONSOLE}
uses
  SysUtils;

const
  DICT_SIZE = 237000;
  MEMO_SIZE = 100000;

type
  tDict = array[1 .. DICT_SIZE] of string;
  tmemoized = array[1 .. MEMO_SIZE] of record
                key, value : string;
              end;

var
  myFile : TextFile;
  dict : tDict;
  dictSize : integer;
  memoized : tmemoized;
  memoSize : integer;
  segment : string;
  searchString : string;

procedure readDict();
var
  i : integer;
begin
  assignFile(myFile, 'DICT.TXT');
  reset(myFile);
  readln(myFile, dictSize);
  for i := 1 to dictSize do
    readln(myFile, dict[i]);
  closeFile(myFile);
end;

procedure readSearchString();
begin
  assignFile(myFile, 'IN.TXT');
  reset(myFile);
  readln(myFile, searchString);
  closeFile(myFile);
end;

procedure writeSegment(s : string);
begin
  assignFile(myFile, 'OUT.TXT');
  rewrite(myFile);
  if (s <> '') then
    writeln(myFile, s)
  else
    writeln(myFile, -1);
  closeFile(myFile);
end;
```

```
function contains(s : string) : boolean;
var
  min, mid, max : integer;
  found : boolean;
begin
  min := 1;
  max := dictSize;
  found := false;

  while (max >= min) AND NOT(found)  do
    begin
      mid := (min + max) div 2;
      if (dict[mid] < s) then
        min := mid + 1
      else if (dict[mid] > s) then
        max := mid - 1
      else
        found := true;
    end;
  if (found) then
    result := true
  else
    result := false;
end;

function containsKey(s : string) : boolean;
var
  i : integer;
begin
  for i := 1 to memoSize do
    if (memoized[i].key = s) then
      begin
        result := true;
        exit;
      end;
  result := false;
end;

function getValue(s : string) : string;
var
  i : integer;
begin
  for i := 1 to memoSize do
    if (memoized[i].key = s) then
      begin
        result := memoized[i].value;
        exit;
      end;
  result := '';
end;

procedure putValue(k : string; v : string);
begin
  inc(memoSize);
  memoized[memoSize].key := k;
  memoized[memoSize].value := v;
end;
```

```
function segmentString(s : string) : string;
var
  i, len : integer;
  prefix, suffix, segSuffix : string;
begin
  if (contains(s)) then
    begin
      result := s;
      exit;
    end;

  if (containsKey(s)) then
    begin
      result := getValue(s);
      exit;
    end;

  len := length(s);
  for i := 2 to len do
    begin
      prefix := copy(s, 1, i-1);
      if (contains(prefix)) then
        begin
          suffix := copy(s, i, len);
          segSuffix := segmentString(suffix);
          if (segSuffix <> '') then
            begin
              putValue(s, prefix + ' ' + segSuffix);
              result := prefix + ' ' + segSuffix;
              exit;
            end;
        end;
    end;
  putValue(s, '');
  result := '';
end;

begin
  memoSize := 0;
  readDict();
  readSearchString();
  segment := segmentString(searchString);
  writeSegment(segment);
end.
```