```
PROGRAM MasterMind;
(*********************************************
 *                                           *
 *   Solution to part B of Trabalho Pratico  *
 *   Introducao a Computacao, 2005/2006      *
 *            - P. Stallinga, 13 June 2006   *
 *                                           *
 *********************************************)

Var code: array[1..20, 1..10] of char;
    colcorr, placecorr: array[1..20] of integer;
    repetition: boolean;
    numcolumns, numcolors, numtry: integer;

FUNCTION CheckIfPossible(n: integer): boolean;
(*****************************************
   Checks if last generated code (n) is
   consistent with previous codes 1..n-1
 *****************************************)
Var i, j, m, placecorrect, colorcorrect: integer;
    copycode, try: array[1..100] of char;
begin
  CheckIfPossible := TRUE;
  for m := 1 to n-1 do     { check if code[n] is consistent with code[m] }
    begin
      placecorrect := 0;
      colorcorrect := 0;
      for i := 1 to numcolumns do
        begin
          copycode[i] := code[m,i];
          try[i] := code[n,i];
        end;
      for i := 1 to numcolumns do
        if try[i]=copycode[i] then
          begin
            placecorrect := placecorrect + 1;
              { make sure it cannot count anymore for correct color: }
            try[i] := ' ';
            copycode[i] := '.';
          end;
      if (placecorrect<>placecorr[m]) then
        CheckIfPossible := FALSE;
      for i := 1 to numcolumns do
        for j := 1 to numcolumns do
          if try[i]=copycode[j] then
            begin
              colorcorrect := colorcorrect + 1;
                { make sure it cannot count twice for correct color: }
              try[i] := ' ';
              copycode[j] := '.';
            end;
      if (colorcorrect<>colcorr[m]) then
        CheckIfPossible := FALSE;
  end;
end;

PROCEDURE GenerateRandomCode;
(*************************************
   Generate random code
 *************************************)
Var i, j, num: integer;
    c: char;
    allowed: boolean;
begin
  Randomize;
  for i := 1 to numcolumns do
    begin
      repeat
```

```
        num := Random(numcolors);
        c := Chr(num+65);
        allowed := TRUE;
        if NOT repetition then
          for j := 1 to i-1 do
            if code[1,j]=c then
              allowed := FALSE;
      until allowed=TRUE;
      code[1,i] := c;
    end;
end;

PROCEDURE GenerateCode(n: integer);
(*****************************************
   Generate code that is consistent with
   results of previous tried codes (1..n-1)
 *****************************************)
Var i, j, k, num: integer;
    c: char;
    allowed, possible, error: boolean;
    lastchar: char;
begin
  { If first try, try something random: }
  if (n=1) then
    begin
      GenerateRandomCode;
      exit;
    end;
  { Else: Find the first combination possible: }
  allowed := TRUE;          { repetitions or not? }
  possible := FALSE;        { consistent with results of previous guesses? }
  error := FALSE;           { no more codes to try? }
  for i := 1 to numcolumns do
    code[n,i] := 'A';
  lastchar := Chr(64+numcolors);
  { repeat looking for a code until one is found that is allowed }
  { and consistent with previous results. Error is generated when }
  { there are no more possibilities: }
  while NOT ((allowed AND possible) OR error) do
    begin

      { 1: check current code for repetitions: }
      allowed := TRUE;
      if NOT repetition then
        for i := 1 to numcolumns-1 do
          for j := i+1 to numcolumns do
            if code[n,i]=code[n,j] then
              allowed := FALSE;

      { 2: check current code for consistence with previous results: }
      if allowed then
        possible := CheckIfPossible(n);

      { 3: if not a possible code, generate next: }
      if NOT (allowed AND possible) then
        { generate next code }
        begin
          { increase last char, for example: ABCD -> ABCE }
          code[n, numcolumns] := Chr(Ord(code[n, numcolumns])+1);
          { check if 'overflow' occured: }
          for i := numcolumns downto 2 do
            if (code[n, i]>lastchar) then
              { example, ABCG -> ABDA (in case lastchar is 'F') }
              begin
                code[n, i] := 'A';
                code[n, i-1] := Chr(Ord(code[n, i-1])+1);
              end;
          { first char cannot have overflow! we ran out of possibilities!: }
```

```
            if (code[n, 1]>lastchar) then
              error := TRUE;
        end;
    end;
  if error then
    begin
      writeln('You were lying. No combination possible!');
      halt;
    end;
  { If it reached here, it means a possible code was found. }
  { Let's exit and ask the user if it was correct }
end;

FUNCTION Verify(n: integer): boolean;
(*****************************************
   Ask the user about the code guessed
 *****************************************)
begin
  writeln('Correct place: ');
  readln(placecorr[n]);
  Verify := (placecorr[n]=numcolumns);
  writeln('Correct color: ');
  readln(colcorr[n]);
end;

Var c: char;
    i, n: integer;
    won: boolean;
    try: string;

begin
  writeln('Number of columns: ');
  readln(numcolumns);
  writeln('Number of colors: ');
  readln(numcolors);
  writeln('Repetitions allowed (y/n): ');
  readln(c);
  repetition := UpCase(c)='Y';
  writeln('Maxium Tries: ');
  readln(numtry);

  n := 0;
  won := FALSE;
  repeat
    n := n+1;
    GenerateCode(n);
    writeln('Let me try: ');
    for i := 1 to numcolumns do
      write(code[n,i]);
    writeln;
    won := Verify(n);
  until (n=numtry) OR won;
  if won then
    writeln('I won!')
end.
```