Programació 1 – Grau en Enginyeria Informàtica
Escola Politècnica Superior
Segon parcial - 15 de Gener de 2019

DNI:

Cognoms, Nom:

---

1. (2 punts) Donat el següent programa en ANSI C++:

#include <stdio.h>

```
void B(int &x, int &y);
int A(int x, int y);

int main( ){
  int a[2] = {5, 6};
  int c = 1;
  B(a[0], a[1]);
  printf("Main:\n a[0] = %d \t a[1] = %d \t c = %d \n", a[0], a[1], c);
}

void B(int &x, int &y){
  int c;
  c = A(x, y);
  x = 8;
  y = 10;
  printf("B:\n c = %d \n",c);
}

int A(int x, int y){
  x = 2 * x;
  y = 2 * y;
  printf("A:\n x = %d \t y = %d \n", x, y);
  return(x - y);
}
```

Indiqueu els valors mostrats per pantalla. Justifiqueu breument la resposta (no vol dir explicar que fa cada línia de codi).

2. (1 punt) Donat el següent programa en ANSI C++:

```
#define T 4
#define M 20

#include<stdio.h>
#include<string.h>

void move(char p2[M+1], char p1[M+1]);

int main( ){
  char lp[T][M+1]={"abcd", "xyz", "PQ", "12345"};
  int i;

  for (i = 1; i < T; i = i + 1) {
    move(lp[i-1], lp[i]);
  }

  for (i = 0; i < T; i = i + 1) {
    printf("La cadena %d és: %s \n", i, lp[i]);
  }
}

void move(char p2[M+1], char p1[M+1]){
  strcpy(p2, p1);
}
```

on el prototipus de la funció `strcpy` declarat al fitxer de capçalera `string.h` és el següent:

```
strcpy(char desti[M+1], char origen[M+1]);
```

Determineu quins valor escriu a la sortida estàndard l'execució del programa. Explica breument que fa el programa (no vol dir explicar que fa cada línia de codi).

DNI:

Cognoms, Nom:

---

3. (3 punts) Implementeu en el llenguatge de programació ANSI C/C++ una funció que anomenareu `diagonals_igual_columnes` que determini si els valors de les diagonals d'una matriu quadrada d'enters sumen igual que els valors de la primera i última columna. Només cal implementar aquesta funció i si en necessiteu alguna altra al aplicar disseny descendent, no s'han d'implementar les funcions d'entrada de dades a la matriu ni la funció `main`.

La funció tindrà el prototipus següent:

```
#define N 4

bool diagonals_igual_columnes(int m[N][N]);
```

La funció `diagonals_igual_columnes` rebrà com a entrada la matriu `m` de N × N valors enters i retornarà `true`, si la suma dels valors de la diagonal principal més els de la diagonal secundària són iguals a la suma dels valors de la primera i última columna, i `false` en cas contrari. Per aquest exercici considerarem que la primera dimensió de la matriu són les files i la segona les columnes. **Feu un disseny que no depengui de la mida de la matriu quadrada**, és a dir, que no depengui de si N val 4 o 100.

Per exemple, si la matriu d'entrada `m` és la següent:

	0	1	2	3
0	1	-8	2	0
1	24	2	-3	-7
2	-17	8	3	10
3	5	8	-64	4

la funció `diagonals_igual_columnes` retornarà `true` ja que els valors de la diagonal principal (1, 2, 3, 4) sumats als valors de la diagonal secundària (5, 8, -3, 0) sumen igual que la suma de tots els valors de la primera columna (1, 24, -17, 5) i última columna (0, -7, 10, 4). Les diagonals sumen 1 + 2 + 3 + 4 + 5 + 8 + (-3) + 0 = 20 i les dos columnes sumen 1 + 24 + (-17) + 5 + 0 + (-7) + 10 + 4 = 20.

4. (4 punts) Donat un text acabat en punt que únicament conté paraules formades per lletres minúscules de llargada màxima 20 i separades per un o més blancs, es demana que dissenyeu i implementeu en el llenguatge ANSI C++ i utilitzant la tècnica de disseny descendent, un programa que compti quantes paraules apareixen en el text que siguin iguals a la inversa de la paraula precedent en el text.

Per exemple, si el text és el següent:

xyz abzyxefg xyz **zyx** xa xyz **zyx** yz abc **cba** **abc** xcba.

La sortida del programa serà:

`Nombre de paraules que són idèntiques a la inversa de la paraula prèvia = 4`

(són les paraules marcades en **negreta** a l'exemple)

Si el text no conté cap paraula, el programa donarà com a sortida:

`Text buit`

Per resoldre el problema podeu utilitzar les funcions `strlen` i `strcpy` de la llibreria estàndard `string.h` amb els prototipus següents:

```
int strlen (char s[M+1]);
strcpy(char desti[M+1], char origen[M+1]);
```

Finalment, per dissenyar i implementar el programa utilitzareu les funcions i accions ja implementades següents (no cal que les torneu a copiar):

```
#include<stdio.h>
#include<string.h>

#define M 20

void llegir_1a_paraula(char p[M+1], char &dc);
void llegir_seguent_paraula(char p[M+1], char &dc);
void saltar_blancs(char &dc);
void llegir_paraula(char p[M+1], char &dc);
bool darrera_paraula(char p[M+1]);

void llegir_1a_paraula(char p[M+1], char &dc){
  scanf("%c", &dc);
  saltar_blancs(dc);
  llegir_paraula(p, dc);
}

void llegir_seguent_paraula(char p[M+1], char &dc){
  saltar_blancs(dc);
  llegir_paraula(p, dc);
}

void saltar_blancs(char &dc){
  while (dc == ' '){
    scanf("%c", &dc);
  }
}

void llegir_paraula(char p[M+1], char &dc){
  int i = 0;
  while (dc != ' ' && dc != '.' && i < M) {
    p[i] = dc;
    i = i + 1;
    scanf("%c", &dc);
  }
  p[i]='\0';
}

bool darrera_paraula(char p[M+1]){
  if (p[0] == '\0') {
    return(true);
  } else {
    return(false);
  }
}
```

Així doncs, heu de programar l'acció `main` i totes les accions i funcions addicionals que necessiteu per tal de fer un codi clar, entenedor i ben estructurat en base a la tècnica de disseny descendent.