Laboratorio di Sistemi Operativi
06 Febbraio 2018
Compito

Si risponda ai seguenti quesiti, giustificando le risposte.

1. (3 punti) sapendo che il comando `date` produce un output come il seguente:

   ```
   gio  1 feb 2018 15:45:26
   ```

   si completi la pipeline seguente, aggiungendo quanto necessario (al posto dei `...`) per estrarre l'anno:

   ```
   date | tr -s ' ' | ...
   ```

   A cosa serve il comando `tr -s ' '`?

   > Per completare la pipeline ed estrarre l'anno va aggiunto il comando `cut -d' ' -f4`. Il comando `tr -s ' '` serve a comprimere eventuali spazi multipli come per esempio quelli tra `gio` e `1` in modo da non sbagliare il riferimento numerico al campo da estrarre con il parametro `-f` del comando `cut`.

2. (5 punti) si scriva uno script `dim.sh` della shell che prenda come argomento sulla linea di comando il percorso di un file e, se quest'ultimo esiste, è un file regolare ed è leggibile dall'utente, ne stampi a video la dimensione in byte e stampi una serie di asterischi di lunghezza pari alla dimensione.

   Esempio (si supponga che il file indice.txt sia lungo 7 byte):

   ```
   > ./dim.sh indice.txt
   Dimensione di indice.txt: 7 byte
   *******
   ```

```
#!/bin/bash

if test $# -ne 1
then
  echo "Utilizzo: $0 <file>"
  exit 1
fi

if test -f $1 -a -r $1
then
  dim=‘wc -c < $1‘
  echo "Dimensione di $1: $dim byte"

  i=0;

  while test $i -lt $dim
  do
    echo -n '*'
    i=$[$i+1]
  done

  if test $dim -gt 0
  then
    echo
  fi

else
```

```
28   echo "Il file passato da linea di comando deve esistere, essere
        leggibile ed essere regolare."
29   exit 2
30 fi
31
32 exit 0
```

3. (8 punti) scrivere il codice di un programma C che riceva in input una lista di n parole e stampi a video tali parole in ordine lessicografico. Si gestisca a scelta (motivata) il formato di input/output.

   Ad esempio (se `es` è il nome del programma eseguibile):

   ```
   $ ./es l3xixograph 0rd3r WORDS aRe 0rd3r3d bas3d on THE az order of their c0mp0nents
   ```

   un possibile output è:

   ```
   $ 0rd3r, 0rd3r3d, aRe, az, bas3d, c0mp0nents, l3xixograph, of, on, order, THE, their, WORDS
   ```

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

int main(int argc, char** argv)
{
    int i, j;
    char *temp;

    if(argc<2) {
      fprintf(stderr,"Utilizzo: %s stringa1 stringa2 ...\n",argv[0]);
      return 1;
    }

    for(i=1; i<argc-1; ++i){
        for(j=i+1; j<argc ; ++j){
            /* è accettabile anche l'uso di strcmp
             * (che distingue fra maiuscole e minuscole)
             */
            if(strcasecmp(argv[i], argv[j])>0) {
                temp=argv[i];
                argv[i]=argv[j];
                argv[j]=temp;
            }
        }
    }

    printf("\nIn lexicographical order: \n");
    for(i=1; i<argc; ++i){
        puts(argv[i]);
    }
    return 0;
}
```

4. (5 punti) Scrivere l'output del seguente programma C.

```c
int main()
{
    int levels=5;

    for (int i = 0; i < levels; i++) {
        for (int j = 0; j < levels - i; j++){
            printf("-");
        }
        for (int k = 0; k < (2 * i + 1); k++){
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```

```
-----*
----***
---*****
--*******
-*********
```

5. (8 punti) Completare il codice (dove compaiono i `...`) per trovare il numero più grande, utilizzando l'allocazione dinamica della memoria: calloc()

   ricordate che: "calloc() Allocates space for an array elements, initializes to zero and then returns a pointer to memory".

   Sintassi di calloc():

   ```
   ptr = (cast-type*)calloc(n, element-size);
   ```

   Nota bene: al posto di una singola occorrenza dei `...` è possibile inserire più linee di codice.

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int i, num;
    ...

    printf("Enter total number of elements(1 to 100): ");
    ...

    // Allocates the memory for 'num' elements.
    ...

    if(data == ...)
    {
        printf("Error!!! memory not allocated.");
        exit(0);
    }

    printf("\n");

```

```
23      // Stores the number entered by the user.
24      for(i = 0; i < num; ++i)
25      {
26        ...
27      }
28
29      // Loop to store largest number at address data
30      for(i = 0; i < num; ++i)
31      {
32        ...
33      }
34
35      printf("Largest element = ...", *data);
36
37      return 0;
38  }
```

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

int main()
{
    int i, num;
    float *data;

    printf("Enter total number of elements(1 to 100): ");
    scanf("%d", &num);

    // Allocates the memory for 'num' elements.
    data = (float*) calloc(num, sizeof(float));

    if(data == NULL)
    {
        printf("Error!!! memory not allocated.");
        exit(0);
    }

    printf("\n");

    // Stores the number entered by the user.
    for(i = 0; i < num; ++i)
    {
       printf("Enter Number %d: ", i + 1);
       scanf("%f", data + i);
    }

    // Loop to store largest number at address data
    for(i = 1; i < num; ++i)
    {
       // Change < to > if you want to find the smallest number
       if(*data < *(data + i))
           *data = *(data + i);
    }

    printf("Largest element = %.2f", *data);
```

```
    return 0;
}
```

6. (4 punti) Dire se le seguenti operazioni sono corrette oppure errate, e motivare la risposta.

```
1 int c, *pc;
```

- `pc= c;`
- `*pc = &c;`
- `pc = &c;`
- `*pc = c;`

(a) `pc= c;`

Sbagliato: si assegna un valore intero ad una variabile di tipo puntatore ad intero (ovvero, di tipo indirizzo). `c` non rappresenta un indirizzo.

(b) `*pc = &c;`

Sbagliato: si assegna alla variabile puntata da `pc` (di tipo intero) l'indirizzo della variabile `c`.

(c) `pc = &c;`

Corretto: si assegna l'indirizzo della variabile `c` al puntatore `pc`. Dopo tale assegnamento `pc` punterà a `c`.

(d) `*pc = c;`

Corretto: si assegna il valore intero memorizzato nella variabile intera `c` alla variabile di tipo intero puntata da `pc` (supponendo che `pc` sia stata correttamente inizializzata).