Laboratorio di Sistemi Operativi
14 Giugno 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 le ore (ovvero, nell'esempio riportato il numero `15`):

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

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

   > Per completare la pipeline ed estrarre l'anno va aggiunta la pipeline `cut -d ' ' -f5 | cut -d ':' -f1`. 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 completi il ciclo `while` dello script `explore.sh` seguente:

```
1  # imposta i separatori interni alla stringa vuota in modo da
       preservare i newline nelle command substitution
2  IFS=
3  lista_file=‘find $1‘ # lista_file conterra‘ tante righe quante sono
4                       # quelle prodotte dal find
5  l=‘echo $lista_file | wc -l‘ # calcola il numero di elementi (righe)
6                               # di lista_file
7  i=1
8  while test $i -le $l;
9  do
10   f=‘echo $lista_file | ...‘
11   echo -n $f '--> '
12   ...                      # qui possono esserci piu‘ linee di codice
13 done
```

   in modo che lo script stampi per ogni elemento in `lista_file`, l'elemento stesso (il pathname), una freccia composta dai caratteri `-->` e la scritta `directory` se si tratta di una directory oppure `file` in caso contrario. Ad esempio:

```
$ ./explore.sh ./concurrent_programming/
./concurrent_programming/prenotazioni.txt --> file
./concurrent_programming/acme_thread.exe --> file
./concurrent_programming/acme_thread.c --> file
./concurrent_programming/ --> directory
```

```
1  IFS=
2  lista_file=‘find $1‘
3  l=‘echo $lista_file | wc -l‘
4  i=1
5
6  while test $i -le $l;
7  do
8    f=‘echo $lista_file | tail -n $i | head -1‘
9    echo -n $f '--> '
10   if test -d $f;
```

```
  then
    echo 'directory'
  fi
  if test -f $f;
  then
    echo 'file'
  fi
  i=$[$i + 1]
done
```

3. (8 punti) Scrivere il codice di un programma C che riceva in input il percorso di un file e generi un secondo file che contenga lo stesso contenuto del primo file a cui viene applicato il seguente filtro: devono essere rimossi dal testo del primo file tutti i caratteri eccetto le lettere dell'alfabeto (maiuscole e minuscole). Si gestiscano inoltre gli eventuali errori (numero di argomenti errato, file non leggibile, ecc.).

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

#define LEN 1024

int main(int argc, char **argv) {
  char line[150];
  int i, j;
  FILE *f,*f2;

  if(argc<2) {
    fprintf(stderr,"Use: filter filename\n");
    return 1;
  }

  f=fopen(argv[1],"r");

  if(f==NULL) {
    fprintf(stderr,"Error opening %s\n",argv[1]);
    return 2;
  }

  f2=fopen("secondo_file","w");

  if(f2==NULL) {
    fprintf(stderr,"Error opening second file\n");
    return 2;
  }

  while(fgets(line,LEN-1,f)!=NULL) {
    for(i = 0; line[i] != '\0'; ++i) {
      while (!( (line[i] >= 'a' && line[i] <= 'z') ||
        (line[i] >= 'A' && line[i] <= 'Z') || (line[i] == '\0'))) {
        for(j = i; line[j] != '\0'; ++j) {
          line[j] = line[j+1];
        }
        line[j] = '\0';
```

```
      }
    }
    fputs(line,f2);
  }

  fclose(f2);
  fclose(f);
  return 0;
}
```

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

```
#include <stdio.h>
int main()
{
    int levels=5;

    for (int i = levels-1; i >= 0; 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 seguente codice che fa uso di allocazione dinamica della memoria.

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

struct course
{
  int marks;
  char subject[30];
};

int main()
{
  struct course *ptr;
  int i, noOfRecords;
  printf("Enter number of records: ");
  scanf(...);
```

```
16
17   // Allocates the memory for noOfRecords structures with pointer
18   // ptr pointing to the base address.
19   ...
20
21   for(i = 0; i < noOfRecords; ++i){
22     printf("Enter name of the subject and marks respectively:\n");
23     scanf("%s %d", ..., ...);
24    }
25
26   printf("Displaying Information:\n");
27
28   for(i = 0; i < noOfRecords ; ++i)
29     printf("%s\t%d\n", ..., ...);
30   return 0;
31 }
```

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

struct course
{
  int marks;
  char subject[30];
};

int main()
{
  struct course *ptr;
  int i, noOfRecords;
  printf("Enter number of records: ");
  scanf("%d", &noOfRecords);

  // Allocates the memory for noOfRecords structures with pointer
  // ptr pointing to the base address.
  ptr = (struct course*) malloc (noOfRecords * sizeof(struct course));

  for(i = 0; i < noOfRecords; ++i){
    printf("Enter name of the subject and marks respectively:\n");
    scanf("%s %d", &(ptr+i)->subject, &(ptr+i)->marks);
  }

  printf("Displaying Information:\n");

  for(i = 0; i < noOfRecords ; ++i)
    printf("%s\t%d\n", (ptr+i)->subject, (ptr+i)->marks);
  return 0;
}
```

6. (4 punti) Completare il seguente codice che fa uso di puntatori.

```
1 #include <stdio.h>
2 int main(){
3    int* pc;
```

```
4     int c;
5     c=22;
6     printf("Indirizzo di c:%u\n", ...);
7     printf("Valore di c:%d\n\n", ...);
8     pc=&c;
9     printf("Indirizzo memorizzato nel puntatore pc:%u\n", ...);
10    printf("Valore puntato da pc:%d\n\n", ...);
11    c=11;
12    printf("Indirizzo memorizzato nel puntatore pc:%u\n", ...);
13    printf("Valore puntato da pc:%d\n\n", ...);
14    *pc=2;
15    printf("Indirizzo di c:%u\n", ...);
16    printf("Valore di c:%d\n\n", ...);
17    return 0;
18 }
```

```
#include <stdio.h>
int main(){
   int* pc;
   int c;
   c=22;
   printf("Address of c:%u\n",&c);
   printf("Value of c:%d\n\n",c);
   pc=&c;
   printf("Address of pointer pc:%u\n",pc);
   printf("Content of pointer pc:%d\n\n",*pc);
   c=11;
   printf("Address of pointer pc:%u\n",pc);
   printf("Content of pointer pc:%d\n\n",*pc);
   *pc=2;
   printf("Address of c:%u\n",&c);
   printf("Value of c:%d\n\n",c);
   return 0;
}
```