Sistemi di Calcolo (A.A. 2025-2026)
Corso di Laurea in Ingegneria Informatica e Automatica
Sapienza Università di Roma

A

Compito (16/10/2026) – Durata 1h 30'

Inserire nome, cognome e matricola nel file `studente.txt`.
ISTRUZIONI PER STUDENTI DSA: svolgere a scelta due parti su tre.

Parte 1 (programmazione IA32)

In questo esercizio, si chiede di tradurre in assembly una funzione che copia un array dato in input in uno spazio di memoria preallocato, fornito anch'esso in input, applicando la funzione compute agli elementi da copiare, solamente se richiesto. Nella directory `E1`, si traduca in assembly IA32 la seguente funzione C scrivendo un modulo `e1B.s`:

```
char* cond_compute(char* array, unsigned* cond,
                   int n, char* out)
{
    int i = 0;
    char res;

    for (i = 0; i < n; i++) {
        out[i] = array[i];
        if (cond[i]) {
            compute(out[i], &res);
            out[i] = res;
        }
    }

    return out;
}
```

L'unico criterio di valutazione è la correttezza. Generare un file eseguibile `e1A` con `gcc -m32 -g`. Per i test, compilare il programma insieme al programma di prova `e1A_main.c`.

Non modificare in alcun modo `e1A_main.c`. Prima di tradurre il programma in IA32 si suggerisce di scrivere nel file `e1A_eq.c` una versione C equivalente più vicina all'assembly.

Parte 2 (programmazione di sistema POSIX)

Si scriva una funzione per la ricerca ed esportazione dei contatti di una rubrica telefonica. Nello specifico, si scriva nel file `E2/e2B.c` una funzione con il seguente prototipo:

```
int cercaEdEsporta(const char * filenameSource, const char * prefix, const char * filenameDest)
```

che, dato in ingresso il nome `filenameSource` del file contenente i dati di una rubrica telefonica ed una stringa `prefix`, scriva nel file con nome `filenameDest` tutti i contatti aventi come prefisso nel `nomecompleto` la stringa `prefix` e restituisce il numero di contatti SCARTATI come valore di ritorno.

I contatti sono salvati su un file testuale `filenameSource`, un record per riga, nel seguente formato:

Descrizione	`nomecompleto`	`numero`
Dimensione	30 bytes	14 bytes

Non vi sono separatori tra i campi di un record in `filenameSource`. I bytes in eccesso sono costituiti da padding rappresentato con il carattere `'_'`.
I record filtrati devono essere riportati sul file `filenameDest`, un record per riga, nel formato

```
nomecompleto,numero
```

SENZA i caratteri di padding `'_'` e nell'ordine INVERSO di apparizione in `filenameSource`.

Ad esempio, assumendo il seguente contenuto di `filenameDest`

```
Mario Rossi____________________+393297854258_
Martina Bianchini_____________+398452158____
```

La funzione `cercaEdEsporta` avente come parametri `prefix = "Mar"` e `filenameDest = "result.csv"` riporterà nel file `result.csv` il seguente contenuto

```
Martina Bianchini,+398452158
 Mario Rossi,+393297854258
```

e darà `0` come valore di ritorno.
Si può assumere che il file `filenameSource` contenga al più 32 record.

Per i test, compilare il codice prodotto insieme al programma di prova `e2A_main.c` fornito, che **non** deve essere modificato. Nota: non modificare i file `test*.txt` che riportano esempi di file contenenti la lista partecipanti.

Parte 3 (quiz)

Si risponda ai seguenti quiz, inserendo le risposte (A, B, C, D o E per ogni domanda) nel file `e3A.txt`. Una sola risposta è quella giusta. Rispondere E equivale a non rispondere (0 punti).

Domanda 1 (Allineamento)

Si consideri la seguente `struct` C:

```
typedef struct S {
  short v;
  void* w;
  short* x;
  char y;
  int z;
} S;
```

Assumendo un'architettura a 32 bit, qual è l'offset del campo `z`?

A	8	**B**	10
C	14	**D**	16

Motivare la risposta nel file `M1.txt`. **Risposte non motivate saranno considerate nulle**.

Domanda 2 (cache)

Si consideri un sistema con una piccola cache completamente associativa contenente 2 sole linee da 16 byte ciascuna e politica di rimpiazzo LRU. Quanti cache miss vengono generati dal seguente frammento di programma? Si assuma che l'array v sia allineato a un indirizzo multiplo di 16 byte e che la cache inizialmente non contenga alcun blocco di memoria in uso al processo.

```
int v[16];
v[0] = 7;
v[6] = 2;
v[2] = 5;
v[15] = 9;
v[2] = v[8];
v[0] = v[1];
```

A	5	**B**	4
C	3	**D**	2

Motivare la risposta nel file `M2.txt`. **Risposte non motivate saranno considerate nulle**.

Domanda 3 (Layout di memoria)

Si consideri l'esecuzione del seguente frammento di programma:

```
short * info;
void foo(int n) {
        short* p = malloc(n*sizeof(short));
        if (p != NULL) info = p;
}
```

Una sola delle seguenti affermazioni che riguarda la collocazione dei dati in memoria in un programma C in un tipico ambiente Linux è **FALSA**, quale?

A	`*p` è in stack e `*info` può essere in heap	**B**	`*p` può essere in heap e `info` è in data
C	`*p` è in heap e `foo` è in text (code)	**D**	`n` è in stack e `*info` può essere in heap

Motivare la risposta nel file `M3.txt`. **Risposte non motivate saranno considerate nulle**.

Domanda 4 (Puntatori o gdb/valgrind)

Data la seguente funzione C:

```
0: #include <stdio.h>
1: int main() {
2:    int i, n = 5, a[n], s;
3:    for(i=1; i<n; i++) a[i] = 1;
4:    s = sum(a, n); // sum somma gli elementi dell'array
5:    printf("%d\n", s);
6:    return 0;
7: }
```

se analizziamo l'esecuzione del programma sotto Valgrind abbiamo:

```
==3207== Memcheck, a memory error detector
==3207== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et
al.
==3207== Using Valgrind-3.13.0 and LibVEX; rerun with -h for
copyright info
==3207== Command: ./main
==3207==
==3207== Conditional jump or move depends on uninitialised value(s)
==3207==    at 0x4E9A8AA: vfprintf (vfprintf.c:1642)
==3207==    by 0x4EA2EE5: printf (printf.c:33)
==3207==    by 0x1087C5: main (main.c:5)
==3207==
==3207== Use of uninitialised value of size 8
==3207==    at 0x4E9683B: _itoa_word (_itoa.c:179)
==3207==    by 0x4E99EDD: vfprintf (vfprintf.c:1642)
==3207==    by 0x4EA2EE5: printf (printf.c:33)
==3207==    by 0x1087C5: main (main.c:5)
==3207==
==3207== Conditional jump or move depends on uninitialised value(s)
==3207==    at 0x4E96845: _itoa_word (_itoa.c:179)
==3207==    by 0x4E99EDD: vfprintf (vfprintf.c:1642)
==3207==    by 0x4EA2EE5: printf (printf.c:33)
==3207==    by 0x1087C5: main (main.c:5)
==3207==
==3207== Conditional jump or move depends on uninitialised value(s)
==3207==    at 0x4E99FE4: vfprintf (vfprintf.c:1642)
==3207==    by 0x4EA2EE5: printf (printf.c:33)
==3207==    by 0x1087C5: main (main.c:5)
==3207==
==3207== Conditional jump or move depends on uninitialised value(s)
==3207==    at 0x4E9AB1C: vfprintf (vfprintf.c:1642)
==3207==    by 0x4EA2EE5: printf (printf.c:33)
==3207==    by 0x1087C5: main (main.c:5)
==3207==
13
```

Qual è la variabile non inizializzata correttamente?

A	`s`	**B**	`n`
C	`a`	**D**	Nessuna delle precedenti

Motivare la risposta nel file `M4.txt`. **Risposte non motivate saranno considerate nulle**.