Politecnico di Milano - Dipartimento di Elettronica, informazione e Bioingegneria

Prof. Mauro Negri

Fondamenti di Informatica **5 luglio 2014**
II appello

Matricola _____ Cognome_____ Nome_____

Durata prova: 2 ore

Istruzioni

LE RISPOSTE DEVONO ESSERE SCRITTE SU QUESTO PLICO (anche sul retro dei fogli)

Si può utilizzare la matita. Vietato l'uso di cellulari, libri, eserciziari, appunti o calcolatrici.

Non è possibile uscire dall'aula durante la prova tranne in casi eccezionali.

La copiatura accertata di parti del compito può comportare la valutazione riprovato dei compiti coinvolti

Esercizio 1.a (10 punti) _____

Esercizio 1.b (4 punti) _____

Esercizio 1.c (6 punti) _____

Esercizio 2 (5 punti) _____

Esercizio 3 (5 punti) _____

Punteggio totale _____

Esercizio 1 (sul retro primo foglio)
Siano date le seguenti definizioni globali:

1. #define maxstring 100
2. char *cost[10] = {"elem1", "elem4", ""}; //esempio di possibile definizione del vettore
3. typedef struct { int campo1; char campo2[maxstring];} t_rec;
4. struct elem { char campo2[maxstring]; int count; struct elem * next;};

e la seguente definizione che si supponga essere inserita nel main che non deve essere implementato.

5. struct elem *primo, *ultimo;

In particolare, la variabile primo (ultimo) contiene l'indirizzo del primo (ultimo) elemento della lista e tali indirizzi corrisponderanno a NULL se la lista è vuota.

parte 1.a Si scriva una funzione chiamata **lettura** che riceve come parametro la lista e il nome di un file che memorizza dei record di tipo t_rec (nessuna ipotesi si può fare sulla quantità di record presenti nel file e sull'ordinamento dei record nel file). La lista ricevuta può trovarsi in qualsiasi stato. La funzione legge dal file (a partire dall'inizio) i record uno alla volta con la funzione che ritenete più opportuna; prende il valore di campo2 del record, elimina i blank presenti nella stringa (davanti, in mezzo, in fondo, invocando una funzione trim(char *blankstring, char *cleanedstring)) che si supponga già implementata. Se un elemento della lista contiene nel campo2 la stringa letta dal file e ripulita dai blank la funzione procede a incrementarne il valore di count di un'unità, mentre se non esiste provvede a creare un nuovo elemento dinamico contenente il valore nel campo2 e count=1 e ad inserirlo come ultimo della lista.

Alla fine della lettura la funzione ritorna, come parametri, la lista modificata e 1 se l'operazione ha avuto successo e 0 altrimenti (errori che impediscono di accedere al file o di modificare la lista).

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAXSTRING 100
#define NOMEFILE "appello.bin"
char* cost[10] = {"elem1", "elem4", ""}; //esempio possibile

typedef struct {int campo1; char campo2[MAXSTRING];} t_rec;
struct elem {char campo2[MAXSTRING]; int count; struct elem *next;};

/* funzione lettura */
int lettura(char *nomefile, struct elem** primo, struct elem** ultimo) {
   FILE *fp; t_rec r; char cleanedstring[MAXSTRING];

  //apertura del file in lettura
  fp = fopen(nomefile, "r");
    if (fp == NULL) {printf("Errore lettura file\n");
return 0;}
  fread(&r, sizeof(r), 1, fp);
  while(!feof(fp)) {
    trim(r.campo2, cleanedstring);
    //chiamata a funzione per aggiornare lista
    if (inserisci(primo, ultimo, cleanedstring) == 0)return 0;
    fread(&r, sizeof(r), 1, fp);
   }

   //stampa
   stampa(*primo); fclose(fp); return 1;
}

int inserisci(struct elem** primo, struct elem** ultimo, char s[MAXSTRING]) {
    struct elem *e, *temp, *prec;    int trovato = 0;
    temp = *primo; prec = NULL;

    while (!trovato && temp != NULL) {
     if (strcmp(s, temp->campo2) == 0) {temp->count = temp->count + 1;
```

```
                                        trovato = 1;}
     else {prec = temp; temp = temp -> next;}
    }

    if (!trovato) {
      //crea elemento e inserisci in coda
      e = malloc(sizeof(struct elem));
      if (e == NULL) {printf("out of memory\n");return 0;}

      strcpy(e->campo2, s);  e->count = 1;

      if (prec == NULL) {*primo = e; *ultimo = e;}
      else {//lista con almeno un elemento
           prec->next = e; *ultimo = prec;}
      }
    return 1;
}
```

parte 1.b Si implementi la funzione trim descritta e utilizzata nell'esercizio precedente.
Soluzione
Il prototipo della funzione è trim(char *blankstring, char *cleanedstring) e quindi non riporta alcuna definizione della dimensione massima delle stringhe e quindi si ipotizza che la dimensione di blankstring sia <= dimensione di cleanedstring e che blankstring abbia al proprio interno \0 come terminatore. Se si avessero le dimensioni della stringa si potrebbero aggiungere il controllo per gestire il caso in cui blankstring non contenga \0 e il controllo sulla dimensionalità.

```
void trim(char *blankstring, char *cleanedstring) {
 int i = 0, j = 0;
 while (blankstring[i] != '\0')
     {if (blankstring[i] != ' ') {cleanedstring[j] = blankstring[i]; j++;}
      i++;
     }    //terminatore
 cleanedstring[j] = '\0';
}
```

parte 1.c Definire una funzione **calcola** che riceve la lista e il vettore cost come parametri. Si ipotizzi che il vettore cost contenga un numero di stringhe a priori non noto.
La funzione ordina gli elementi della lista per valori crescenti della stringa campo2, invocando una funzione **Ordina**, che riceve la lista, la ordina sul campo2 e la restituisce ordinata (questa funzione va invocata, MA NON implementata). Poi la funzione scandisce la lista e visualizza a terminale il contenuto di campo2 e count di un elemento della lista se il valore di campo2 è tra quelli presenti nel vettore cost; il vettore cost memorizza una stringa vuota nel primo elemento libero (ad esempio, se contiene tre stringhe esse occuperanno le posizioni 0,1 e 2 e nell'elemento 3 ci sarà una stringa vuota).

```
void calcola (struct elem** primo, struct elem** ultimo, char *cost[10]) {

 struct elem* temp; int trovato, i;

 //ordina la lista
 Ordina(primo, ultimo);

 //scorri lista e stampa gli elementi che compaiono anche in cost
 temp = *primo;
 while (temp != NULL) {
   i = 0; trovato = 0;
   while (!trovato && i < 10 && strcmp(cost[i], "") != 0) {
      if (strcmp(cost[i], temp->campo2) == 0)
       {printf("Trovato in vettore %s con count=%d\n",temp->campo2,temp->count);
        trovato=1;
       }
```

```
      i++;
       }
      temp = temp -> next;
  }
}

/* NON RICHIESTA - SOLO PER INFO*/
void Ordina(struct elem **primo, struct elem **ultimo) {return;}


/* NON RICHIESTA - SOLO PER TEST stampa a video la lista dinamica */
void stampa(struct elem* lista) {
    while (lista != NULL) {
     printf("[%s %d] -> ", lista->campo2, lista->count);lista = lista -> next;
    }
    printf("NULL\n");
}


/* NON RICHIESTA - SOLO PER TEST scrive il file binario da leggere */
void scrivifile(char *nomefile) {
    t_rec r;    int i, max;
    char *s[MAXSTRING] = {"elem1", "elem1", "elem2", "elem3", "elem4", "elem4"};
    int v[] = {1,2,3,4,5,6};  FILE *fp;
    max = 6;

    fp = fopen(nomefile, "w");
    if (fp == NULL) {printf("errore scrittura file\n"); return;}

    for (i = 0; i < max; i++) {
        printf("%s %i\n", s[i], v[i]); r.campo1 = v[i];strcpy(r.campo2, s[i]);
        fwrite (&r, sizeof(r), 1, fp);
    }
    fclose(fp);
}

int main() {
    struct elem *primo, *ultimo;  primo = NULL; ultimo = NULL;

    scrivifile(NOMEFILE);
    lettura(NOMEFILE, &primo, &ultimo);
    calcola(&primo, &ultimo, cost);
    return 0;
}
```

Esercizio 2.

1. Indicare la rappresentazione in complemento a 2, utilizzando 8 bit, dei numeri:
   A = 55 (base 10) 00110111
   B = -67 (base 10) 01000011 -> 10111101

2. Data la definizione della variabile float v= - 0,777 descrivere la sua rappresentazione in virgola mobile in base allo standard IEEE754 semplice precisione FP32 descritto a lezione;
   a) Definire prima il numero nel formato normalizzato intermedio (TRONCARE parte decimale alla terza cifra decimale):

$$-1{,}554 * 2^{-1}$$

   b) Codificare poi il numero normalizzato usando il formato FP32:
   S (1 bit di segno): 1
   M (23 bit per la mantissa – fermarsi ai primi 6 bit): 100011
   E (8 bit per l'esponente): 01111110

c)Indicare quale sia il vero valore (in base 10) descritto dalla sequenza di bit memorizzata in S, M ed E del punto b) precedente (si consiglia di esprimere la mantissa in termini frazionari).

-(1+ ½+ 1/32 +1/64)= 0,773

3. Data la rappresentazione IEEE754 - FP32 indicare quali siano gli estremi dell'intervallo dei valori rappresentabili (dare una breve motivazione).

   $-3{,}4*10^{38}$ …$-1{,}1*10^{-38}$ (FLT_MIN) $1{,}1*10^{-38}$… $3.4*10^{38}$ (FLT_MAX)

   FLT_MIN = $(1+0)2^{-126}$ FLT_MAX = $(1+0{,}5+0{,}25+\ldots)2^{127} = 2^{128}$

4. Tradurre il seguente programma C in assembler

```
…
int  V [10], C;
int F(int *p1, int p2)
    {int S[3];….}
int main ()
    { V[2]=5;
      C=F(V, C);
      }
```

V: .RES 10
C:.RES 1
STACK: .RES 1000
INIZIO: MOV #STACK, SP
ADD #999, SP
 	
MOV #V, R1
ADD #2, R1
MOV #5, (R1)
 	
ADD #-1,SP
MOV #V, (SP)
ADD #-1,SP
MOV C, (SP)
ADD #-1,SP
JSR F
	ADD #3, SP
	MOV (SP), C
	EXIT
	
	F: MOV R0, (SP)
	ADD #-1,SP
	MOV SP, R0
	ADD #-3, SP
	…
	ADD #4, SP
	MOV (SP), R0
	RTS
	.END INIZIO

Esercizio 3.

Si scriva un programma C eseguito da un processo (padre), il quale genera tre processi figli e poi aspetta finché ha terminato il primo processo generato e solo a questo punto termina l'esecuzione indipendentemente dall'esistenza di uno degli altri processi generati ancora in esecuzione.

Ogni processo figlio lancia in esecuzione un programma di nome PROG passando come parametri le stringhe *input* e *output* e infine eseguendo una exit().

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
int main ()
{pid_t pid[3]; int status, i;

for (i =0;i<=2;i++)
 {pid[i]=fork();
     if (pid[i]==-1) {printf("\nan error occurred executing fork\n"); exit(0);}
  if (pid[i]==0)
     {execl("./prog",input, output, NULL);
      exit(0);  // eseguita se execl fallisce
     }
```

```
 }
// parte solo per il padre
waitpid(pid[0], &status, 0);
exit(1);
}
```