Politecnico di Milano - Dipartimento di Elettronica, informazione e Bioingegneria

Prof. Mauro Negri

Fondamenti di Informatica **4 marzo 2014**
I appello

Matricola _____ Cognome_____ Nome_____

Durata prova: 2 ore

Istruzioni

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

Non separare questi 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 comporterà in base alla gravità l'annullamento o la valutazione riprovato dei compiti coinvolti

Esercizio 1.a (10 punti) _____

Esercizio 1.b (10 punti) _____

Esercizio 2 (5 punti) _____

Esercizio 4 (5 punti) _____

Punteggio totale _____

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

1. #define maxel 100
2. typedef struct elem { int desc1; char desc2[10];} tipoel;
3. struct eldin { tipoel dato; struct eldin * next;};

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

4. struct eldin *lista;

parte 1.a Si definisca un vettore di al massimo maxel elementi di tipo "tipoel".
Poi si scriva una funzione chiamata **carica** che riceve come parametro il vettore appena definito, due numeri interi A e B e il nome di un file. La funzione legge dal file (a partire dall'inizio del file) i record uno alla volta con la funzione che ritenete più opportuna; il file contiene una sequenza (di dimensione non nota) di record di tipo "tipoel" ordinati per valori crescenti del campo "desc1" e li carica nel vettore a partire dalla posizione A (valore dell'indice nel vettore) e sino alla posizione B che deve essere >= a quella precedente. Alla fine della lettura la funzione ritorna, come parametri, il vettore caricato e la posizione occupata dall'ultimo record caricato nel vettore (qualsiasi ipotesi fatta in merito a quest'ultimo parametro deve essere esplicitata nel testo con un commento). Inoltre sempre come parametro restituisce 1 se l'operazione ha avuto successo e 0 altrimenti. Si noti che i valori dei parametri A e B non sono controllati da chi attiva la funzione pertanto si devono inserire opportuni controlli sui loro valori all'interno della funzione e se tali valori non risultano accettabili la funzione non legge nulla e ritorna 0.

parte 1.b Si scriva una funzione chiamata **DEL** che riceve, come parametro, la lista dinamica definita nel main nella quale si supponga che esistano elementi duplicati (con lo stesso numero nel campo "desc1". Riceve inoltre il vettore di elementi di tipo "tipoel" e due numeri interi A e B come parametri. La funzione elimina fisicamente tutti gli elementi dinamici della lista che contengono nel campo "desc1" uno dei numeri contenuti nel campo "desc1" degli elementi che vanno dalla posizione A (inclusa) alla posizione B (inclusa) del vettore ricevuto; i valori di A e B non vanno controllati perché supposti corretti. La funzione ritorna la lista modificata e il numero di elementi cancellati.

Parte 1.a

```
//vettore globale da definire
tipoel vettore[maxel];
int carica(tipoel v[maxel], int A, int B, char *nomefile, int* pos)
// vettore è sempre passato per indirizzo, pos rappresenta la posizione dell'ultimo elemento //caricato con l'ipotesi che se non carico nulla valga ad esempio -1 per distinguere questo //caso dagli altri.
{    FILE *fp;   int i = 0;

 if (A < 0 || A > B || B > maxel - 1){printf("Valori di A e B errati\n"); return 0;}

 fp = fopen(nomefile, "rb");
   if (fp == NULL){printf("Errore durante l'apertura del file\n");return 0;}

 fread(&v[A+i], sizeof(tipoel), 1, fp)
   while(!feof(fp) && (A+i <= B))
   {printf("Inserito elemento in posizione %d\n", A+i);
    i++;
    fread(&v[A+i], sizeof(tipoel), 1, fp)
    }
   fclose(fp);

   if (i > 0) {*pos = A+i-1;} else {*pos = -1;    }
      //se non ho letto nulla, ritorno -1
 return 1;
}
```

```
struct eldin* DEL (struct eldin *lista, tipoel v[maxel], int A, int B,  int* cancellati)
{int i;    struct eldin* tmp, *prec;

  *cancellati = 0;
 for (i=A; i<=B; i++)
    { if (i == A) ||  (v[i].desc1 != v[i-1].desc1)
        {  prec = NULL;    tmp = lista;
           while(tmp != NULL )
            {  if (tmp -> dato.desc1 == v[i].desc1)
              {   if (prec == NULL)   { lista = tmp -> next;  free(tmp);   tmp = lista;}  //caso testa
                   else   {prec -> next = tmp -> next;  free(tmp); tmp = prec -> next; }
                  printf("Cancellato valore %d\n", v[i].desc1);
                  (*cancellati)++;
               }
                else
                   {  prec = tmp;    tmp = tmp -> next;   }    //avanzo puntatore nella lista
            }
         }
      }
    return lista;
}
```

Esercizio 2.

1. Indicare la rappresentazione in complemento a 2, utilizzando 8 bit, dei numeri:
   A = 77 (base 10) 0100 1101
   B = -76 (base 10) 1011 0100 <- 0100 1100
2. Eseguire la somma A+B in binario dei numeri interi A=200 e B=56, utilizzando 9 bit per la rappresentazione. Mostrare il risultato in binario generato dall'unità aritmetico-logica; tradurre in decimale il numero binario ottenuto, riportare il valore del bit di carry (riporto) e di overflow dopo l'operazione, indicando in breve il perché del valore di carry e overflow.

   011001000 200
   000111000 56

   100000000 Decimale corrispondente: -256
   bit carry: 0
   bit overflow: 1

Motivazione:
Con 9 bit ci sono 512 configurazioni per l'intervallo [-256, 255]. I due numeri sono quindi rappresentabili, ma non la loro somma. Pertanto il numero generato risulta essere il numero più grande negativo che rappresenta il successivo del numero più grande positivo.
Carry 0 in quanto non c'è riporto sul primo bit a sinistra
Overflow 1 perché operandi con segno concorde producono somma con segno discorde.

3. Data la definizione della variabile float v= 31 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):

      $1{,}937*2^4$

   b) Codificare poi il numero normalizzato usando il formato FP32:
      S (1 bit di segno): 0
      M (23 bit per la mantissa – fermarsi ai primi 6 bit): 111011
      E (8 bit per l'esponente): 1000 0011

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).

16 (1+1/2 + 1/4 + 1/8 + 1/32 + 1/64) =123/4 = 30,75

4. Data la rappresentazione IEEE754 - FP32 del numero 33 indicare quale sia il primo numero decimale (esprimerlo eventualmente come formula) rappresentabile successivo al numero 33.

   33 corrisponde a 1,… * 2^5 pertanto il numero successivo si ottiene incrementando il 23esimo bit della mantissa di un'unità che vale 2^{-23} , pertanto il numero successivo sarà 33+ 2^{-23} *2^5

5. Supponendo di dover tradurre il seguente programma C in assembler disegnare il record di attivazione (contenuto di ogni parola allocata, indirizzo contenuto nei registri utilizzati) nel momento in cui la funzione fun comincia ad eseguire le proprie istruzioni.

```
#include <stdio.h>
struct rec {int desc1; int desc2;};
struct rec miorec; int val;
int fun(struct rec t, int *a)
    {int vet[3];….}
int main ()
    { …
      c=fun(miorec, &val);
      }
```

	<- (SP)
Vet[3] locale	
Vet[2] "	
Vet[1] "	<-(R0)
Valore R0 salvato	
Indirizzo ritorno codice	
Indirizzo val a	
t.desc2 miorec.desc2	
t.desc1 miorec.desc1	
Risultato per return	

Esercizio 3.

Si scriva un programma C eseguito da un processo (radice), il quale chiede al terminale un valore intero maggiore di zero memorizzato nella variabile "depth". Il processo padre cerca di creare una gerarchia di processi figli con profondità = depth (ad esempio, se depth=3 il padre – primo elemento - genera un processo figlio – secondo elemento -, il quale a sua volta genera un proprio processo figlio – terzo e ultimo elemento). Ogni processo (incluso la radice) dopo aver creato il proprio processo figlio si mette in attesa della sua terminazione. L'ultimo figlio creato termina subito la propria esecuzione ritornando al proprio processo padre un numero corrispondente al proprio livello nella gerarchia (ad esempio, ritorna 3 se depth=3), mentre gli altri processi alla terminazione del proprio figlio visualizzano al terminale il pid del figlio terminato e poi a loro volta terminano l'esecuzione ritornando un numero corrispondente al proprio livello nella gerarchia (il padre della gerarchia restituisce 1, primo figlio creato restituisce 2, il secondo 3 e così via); nel caso in cui un processo fallisca nel creare il proprio figlio allora termina l'esecuzione riportando il codice -1, invece del proprio livello e in questo caso anche gli altri processi ritorneranno -1 invece del proprio livello.

Soluzione:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
int main ()
{pid_t pid; int status, depth, livello=1;

do {printf(" inserire profondità:"); scanf(" %d ",&depth);}  while (depth<=0);

while (depth > livello)                 //controllo per gestire sia depth=1 o l'ultimo figlio generato
      { pid = fork();     if (pid == -1) exit (-1);
          // (else)
          if (pid!=0)
            { pid=wait(&status);
              printf("pid=%d ", pid );
```

```
            if ((status/256) != -1) exit(livello);// ogni padre deve stabilire cosa propagare
           else exit(-1);                          // (livello o errore)
            }
           else livello++;
      }
printf("\nexit livello %d", livello); //per controllo esito esecuzione
exit(livello);
 }
```