COGNOME: **NOME:** **MATRICOLA:**

ESAME (COMPLETO) di Programmazione I, 15 Giugno 2015.

Esercizio 1 [11 punti] Si definisca la seguente funzione

```
int *pari(const int *lista, int n, int *count) { ... }
```

che dato in input un array di interi, restituisce un array contenente solo i numeri pari di `lista` e memorizza il loro conteggio in `count`.
Si definisca il metodo

```
struct list *to_list(const int *array, int n) { ... }
```

che costruisce una lista di numeri contenente i valori di `array` nello stesso ordine con cui sono dati all'interno del vettore.
Inoltre, si definisca la funzione

```
void print(struct list *li) { ... }
```

che stampa il contenuto della lista `li`, in ordine dal primo all'ultimo elemento.
Si implementi un `main` che legga da tastiera 10 numeri e li memorizzi in un array, estragga solo i numeri pari e li inserisca in una lista, usando i metodi descritti sopra, quindi stampi il contenuto della lista.
Se il codice é corretto, allora una possibile esecuzione del programma sarà la sequente:

```
inserisco i numeri 1 2 3 4 5 6 7
i numeri pari sono 2 4 6
```

Esercizio 2 [11 punti] Si descriva, con l'ausilio di una figura e di opportuni commenti, lo spazio in memoria dedicato al programma seguente, evidenziando in particolare i segmenti stack, heap e data nell'istante immediatamente prima dell'esecuzione dell'istruzione *return 0*:

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

int a, b;

int f1(int x, int y){
   return x + y;
}

int main(void){
  int x, y;

  int *p1 = (int *) malloc(sizeof( int ));
  *p1 = f1(a, 17);

  return 0;
}
```

Esercizio 3 [11 punti] Definire le struct `temperatura`, nodo per una lista concatenata di temperature (con valore di tipo double).
Realizzare le seguenti funzioni:
`in_range()` determina se un valore di temperatura (double) appartiene ad un certo intervallo di valori.
`last()` dato un puntatore ad una lista di temperature, scorre tale lista ritornando il puntatore all'ultimo

elemento.
`filter()` dato un array di temperature, costruisce una lista di temperature appartenenti all'intervallo [BEGIN, END] e ritorna il puntatore al primo elemento (head).
Se tutto é corretto, l'esecuzione del sequente programma:

```
const double BEGIN = 4;
const double END = 24;

int main(void){
    float values[] = {3.4, 22.7, -7, 16.4, 21.3};
    struct temperatura *head = filter(values, 7);
    print(head);
}
```

stamperà:

```
22.70 16.40 21.30
```

NOTA: La funzione print() **non** è richiesta.

Soluzione 1

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

struct list{
 int value;
 struct list *next;
};

int *pari(const int *lista, int n, int *count){
 *count = 0;
 int i,j;
 for(i = 0; i < n; i++){
  if(lista[i] % 2 == 0){
   (*count)++;
  }
 }
 int *out = (int *)malloc((*count) * sizeof(int));
 j = 0;
 for(i = 0; i < n; i++){
  if(lista[i] % 2 == 0){
   //*(out + j) = *(lista + i);
   out[j] = lista[i];
   j++;
  }
 }
 return out;
};

//soluzione iterativa
struct list *to_list(const int *array, int n){
 if(n == 0){
  return NULL;
 }
 else{
  struct list *head = (struct list *)malloc(sizeof(struct list));
  head->value = array[0];
```

```
  struct list *tail = head;
  int i;
  for(i = 1; i < n; i++){
   tail->next = (struct list *)malloc(sizeof(struct list));
   tail = tail->next;
   tail->value = array[i];
  }
  tail->next = NULL;
  return head;
 }
};

//soluzione ricorsiva
struct list *rec_to_list(const int *array, int n){
 if(n == 0){
  return NULL;
 }
 else{
  struct list *this = (struct list *)malloc(sizeof(struct list));
  this->value = array[0];
  this->next = rec_to_list(array + 1, n - 1);
  return this;
 }
};

//soluzione ricorsiva 2
struct list *construct_l(int value, struct list *next){
 struct list *this = (struct list *)malloc(sizeof(struct list));
 this->value = value;
 this->next = next;
};
struct list *rec2_to_list(const int *array, int n){
 if(n == 0){
  return NULL;
 }
 else{
  return construct_l(array[0], rec2_to_list(array + 1, n - 1));
 }
};

void print(struct list *li){
 while(li != NULL){
  printf("%i ", li->value);
  li = li->next;
 }
 printf("\n");
};

void destroy(struct list *l){
 if(l != NULL){
  struct list *c = l;
  struct list *p = NULL;
  while(c != NULL){
   if(p != NULL){
    free(p);
   }
   p = c;
```

```
   c = c->next;
  }
  free(p);
 }
};

int main(void){
 int array[10];
 printf("Inserisci 10 numeri:\n");
 int i;
 for(i = 0; i < 10; i++){
  scanf("%i", &(array[i]));
 }
 int count = 0;
 int *pp = pari(array, 10, &count);
 struct list *l = to_list(pp, count);
 //struct list *l = rec_to_list(pp, count);
 //struct list *l = rec2_to_list(pp, count);
 printf("I numeri pari sono %d:\n", count);
 print(l);
 destroy(l);
}
```

Soluzione 2 Si veda l'immagine allegata.

Soluzione 3

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

const double BEGIN = 4;
const double END = 24;

struct temperatura {
    float value;
    struct temperatura *next;
};

bool in_range(double value, double begin, double end) {
    return value >= begin && value <= end;
}

// Soluzione iterativa
struct temperatura* last_i(struct temperatura* head) {
    if(head == NULL) {
        return head;
    } else {
        struct temperatura* temp = head;

        while(temp->next != NULL) {
            temp = temp->next;
        }

        return temp;
    }
}

// Soluzione ricorsiva
```

```
struct temperatura* last_r(struct temperatura* head) {
    if(!head || !head->next) {
        return head;
    } else {
        return last(head->next);
    }
}

struct temperatura* filter(float values[], int size) {
    int i;
    struct temperatura *temp;
    struct temperatura *prev;
    struct temperatura *head = NULL;

    for(i = 0; i < size; i++) {
        if(in_range(values[i], BEGIN, END)) { // in_range_i() oppure in_range_r()
            temp = malloc(sizeof(struct temperatura));
            temp->next = NULL;
            temp->value = values[i];

            if(head == NULL) {
                head = temp;
            } else {
                prev = last(head);
                prev->next = temp;
            }
        }
    }
    free(temp);
    return head;
}
```