Di seguito viene riportato lo svolgimento di una simulazione d'esame, con l'implementazione guidata svolta in aula e la relativa discussione in forma di commenti.

TIRO AL BERSAGLIO

Il testo della prova d'esame è disponibile sulla piattaforma Kiro, sotto le cartelle:
per gli iscritti a Bioingegneria
`FondInf-bioing > Tutorato > esame bersagli > bersaglio.pdf`
per gli Elettronici-Informatici
`FdI 2016-17 > Tutorato 12 - "Bersaglio" > Traccia tutorato`.

BERSAGLIO1.C

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

#define DIM_NOME 120 //buffer di dim ragionevole
#define DIM_BUFF 120
#define DIM_PUNTI 15 //numero di tiri di ogni giocatore

struct giocatore {
  char nome[DIM_NOME];
  int dist[DIM_PUNTI];
};

struct giocatore *caricaFile(FILE *fin, int *tot){

  int i=0,j,dim = 10; //dimensione di partenza
  char buff[DIM_BUFF]; //buffer per sscanf e fgets
  struct giocatore *vett;

  /*
      alloco esattamente "dim" * dimensione_di(struct giocatore)
  */
    vett = malloc(dim*sizeof(struct giocatore));

  while(fgets(buff,sizeof(buff),fin) != NULL){
    strcpy(vett[i].nome,buff);

    for(j=0;j<DIM_PUNTI;j++) {
      fgets(buff,sizeof(buff),fin);
      sscanf(buff,"%d", &vett[i].dist[j]);
    }

    i++; //avanziamo nello spazio nella malloc

    if(i >= dim){
      //se "dim" non basta, lo raddoppio
      dim = dim*2;

      //rialloco il vecchio "vett" così "vett" è grande il doppio
      vett = realloc(vett, dim*sizeof(struct giocatore));
    } //fine if
  }
  *tot = i;
  return vett;
}
```

```c

void maxCentri(struct giocatore *giocatori, int tot, int *tot_bombers){

  int i, j, max=0;
  int bombers[tot];
  int num_centri;

    // contiamo ogni giocatore quanti centri ha fatto
  for(i=0;i<tot;i++){
    num_centri = 0;
    for(j=0;j<DIM_PUNTI;j++){
      if(giocatori[i].dist[j]<= 30)
        num_centri++;
    }
    bombers[i]= num_centri;
  }

  max=bombers[0];

    // chi ha fatto più centri
  for(i=0;i<tot;i++)
    if(bombers[i] >= max)
        max = bombers[i];

  j=0;
  printf("[MAXCENTRI]\n");
  for(i=0;i<tot;i++)
    if(bombers[i]==max) {
      printf("%s",giocatori[i].nome);
      j++;
    }

  *tot_bombers = j;
}

void dist_min_max(struct giocatore *g, int tot, int *dist_min, int *dist_max)
{
  int i, j;

  *dist_min = g[0].dist[0];
  *dist_max = g[0].dist[0];

  /*
    Cicliamo su ogni giocatore ed i suoi tiri
    applicando l'algoritmo solito di ricerca min/max
  */

  for (i = 0; i < tot; i++) {
    for (j = 0; j < DIM_PUNTI; j++) {
      if (g[i].dist[j] < *dist_min)
        *dist_min = g[i].dist[j];
      if (g[i].dist[j] > *dist_max)
        *dist_max = g[i].dist[j];
    }
  }
}

int main(int argc, char **argv) {

    /* //debug
```

```
106     int i,j;
107     printf("#numero arg: %d\n",argc);
108     for(i=0;i<argc;i++)
109     printf("#argv[%d]: \"%s\"\n", i, argv[i]);
110         */
111
112   int tot; //tot giocatori
113   int tot_bombers =0; // tot campioni
114   FILE *fin; //ptr al file
115   struct giocatore *giocatori;
116
117   fin = fopen(argv[1],"r"); //apro file
118   giocatori = caricaFile(fin,&tot); //funzione carica
119   fclose(fin); //chiudo
120
121   /* //debug
122   for(i=0; i<tot; i++) {
123     printf("#%s#",giocatori[i].nome);
124     for(j=0;j<DIM_PUNTI;j++)
125       printf("%d ",giocatori[i].dist[j]);
126       printf("\n");
127   }
128    */
129
130   maxCentri(giocatori,tot, &tot_bombers);
131
132   int dist_min, dist_max;
133   dist_min_max(giocatori, tot, &dist_min, &dist_max);
134   printf("[MINMAX]\n%d\n%d\n", dist_min, dist_max);
135
136   return 0;
137 }
```