Esercitazione 6 soluzioni

Esercizio 5:

```
#include <stdio.h>

char primaMaiuscola(char vett[], int dim) {
  int i=0;
  char ret = '\n';
  /* in questo caso vale \n sia come ritorno di default che come       controllo
per verificare se e' stata trovata una maiuscola.
      Scorro il vettore con uno while poiche' potrei dovermi fermare
      prima della fine.
  */
  while(i<dim && ret=='\n') {
    if('A' <= vett[i] && vett[i] <= 'Z')
      ret = vett[i];
    i++;
  }
  return ret;
}

#define LUNG1 6
#define LUNG2 4

void main() {
  int i;
  char vet1[LUNG1]= {'g', 'f', 'm', 'D', 'h', 'E' };
  char vet2[LUNG2]= {'r', 'w', 's', 'a'};

  printf("vet1 = [ ");
  for(i=0; i<LUNG1; i++)
    printf("%c ", vet1[i]);
  printf("]\n");

  printf("vet2 = [ ");
  for(i=0; i<LUNG2; i++)
    printf("%c ", vet2[i]);
  printf("]\n");


  printf("La prima maiuscola di vet1 e': %c\n", primaMaiuscola(vet1, LUNG1));
  printf("La prima maiuscola di vet2 e': %c\n", primaMaiuscola(vet2, LUNG2));

}
```

Esercizio 7:

Scrivere una funzione somma_array, che sommi due vettori di interi, inizializzati prendendo valori inseriti in input dall'utente.

```
#include <stdio.h>

char somma_array(int vett1[], int vett2[]) {
  int a[10], b[10], c[10];
int i,n;
```

Esercizio 13:

```
#include <stdio.h>

void dollarize(char arr[], int dim) {
  int i;
  char el;
  for(i=0; i<dim; i++) { /* scorro tutto il vettore */
    el = arr[i];
    if('A'<=el && el<='Z') /* trasformo le maiuscole in minuscole */
      el = el - 'A' + 'a';

    if(el=='a' || el=='e' || el=='i' || el=='o' || el=='u') {
      /* se incontro una vocale, la sostituisco */
      arr[i]='$';
    }
  }
}

#define LUNG 6

void main() {

  char vet[LUNG]= {'g', 'a', 'm', 'D', 'U', 'e' };
  int i;

  printf("vet = [ ");
  for(i=0; i<LUNG; i++)
    printf("%c ", vet[i]);
  printf("]\n");

  dollarize(vet, LUNG);

  printf("vet = [ ");
  for(i=0; i<LUNG; i++)
    printf("%c ", vet[i]);
  printf("]\n");

}
```

Esercizio 16:

```
#include <stdio.h>

/* funzione ausiliaria per ottenere il valore minimo fra due */
int minimo(int a, int b) {
  if(a<b) return a;
  return b;
}

/*
Notate che il limite superiore di ricerca viene calcolato con l'uso della
funzione ausiliaria "minimo" prima definita.
 */
int coprimi(int a, int b) {
  int i, ret=1, upperLimit;

  /* se il minimo fra a e b e' pari, upperLimit e' la sua meta' precisa.
```

```
  Se e' dispari, la sua meta' approssimata per eccesso. */
  upperLimit = (minimo(a,b)+1) / 2;

  /* intervallo [2, N/2] */
  i=2;
  while(i<=upperLimit && ret==1) {
    /* se i due numeri sono entrambi divisibili per i, non sono coprimi */
    if(a%i==0 && b%i==0)
      ret = 0;

    i++;
  }
  return ret;
}
```

Esercizio 17:

```
int arrayDiCoprimi(int vett[], int dim) {
  int i, j;
  int ret = 1;
  for(i=0; i<dim; i++) {
    for(j=i+1; j<dim; j++) {
      /* l'esercizio vuole verificare se tutti gli elementi dell'array
      sono coprimi tra loro: la proprieta' deve essere vera per ogni
      coppia di elementi e quindi i risultati delle chiamate alla
      funzione vanno messi in AND */
      ret = ret && coprimi(vett[i], vett[j]);
    }
  }
  return ret;
}

#define LUNG 4

void main() {

  int vet[LUNG] = {5,18,17,7}; /* tutti coprimi */
  int vet2[LUNG] = {5,18,17,15}; /* non tutti coprimi */
  int i;

  printf("vet = [ ");
  for(i=0; i<LUNG; i++)
    printf("%d ", vet[i]);
  printf("]\n");

  if(arrayDiCoprimi(vet, LUNG))
    printf("L'array contiene solo numeri coprimi\n");
  else
    printf("L'array non contiene solo numeri coprimi\n" );

  printf("vet2 = [ ");
  for(i=0; i<LUNG; i++)
    printf("%d ", vet2[i]);
  printf("]\n");

  if(arrayDiCoprimi(vet2, LUNG))
    printf("L'array contiene solo numeri coprimi\n");
  else
```

```
    printf("L'array non contiene solo numeri coprimi\n" );

}
```

Esercizio 18:

```
#include <stdio.h>

void ordina(int vett[], int dim) {
  int act, cur, min, pos;

  for(act=0; act<dim; act++) {
    /* ad ogni iterazione, l'elemento attuale e' il minimo iniziale */
    min = vett[act];
    pos = act;

    /* a partire dalla posizione successiva a quella attuale, cerco il minimo */
    for(cur=act+1; cur<dim; cur++) {
      if(vett[cur]<min) {
        min = vett[cur];
        pos = cur;
      }
    }

    /* se ho trovato un elemento minimo diverso da quello attuale, lo sposto
    al posto di quello attuale. Notate che lo swap si puo' eseguire senza
    variabile temporanea aggiuntiva perche' ho il valore di min salvato
    in precedenza. */
    if(pos>act) {
      vett[pos]=vett[act];
      vett[act]=min;
    }
  }
}

#define LUNG 11

/* un semplice main che crea un array, lo stampa a video
lo passa alla procedura e quindi stampa a video l'array modificato */
main() {

  int vet[LUNG] = {5,6,1,3,4,5,8,2,1,1,6};
  int i;

  printf("vet = [ ");
  for(i=0; i<LUNG; i++)
    printf("%d ", vet[i]);
  printf("]\n");

  ordina(vet, LUNG);

  printf("vet = [ ");
  for(i=0; i<LUNG; i++)
    printf("%d ", vet[i]);
  printf("]\n");


}
```

Esercizio 19:

```
#include <stdio.h>
main()
{ int i,j,max,temp, a[10];
for(i=0; i<=9; i++)
scanf("%d", &a[i]);
trovaMax(a)
}

void trovaMax(int *array[])
{
for(i=0; i<=2; i++)
{ max = i;
for(j=i+1; j<= 9; j++)
if(a[j]>a[max])
max=j;
temp= a[max];
a[max]=a[i];
a[i]= temp;
}
printf("questi sono gli elementi dell'array con\n i tre
numeri piu` grandi nelle prime tre posizioni:\n");
for(i=0; i<=9; i++)
printf("%d\n", a[i]);
}
```