PROBLEMA 1

Progettare e codificare in C una procedura che dati tre interi calcoli contemporaneamente il massimo e il minimo.

PROBLEMA 2

Progettare e codificare un programma C che dato un numero N dica se e' un quadrato perfetto e in quel caso ne restituisca la radice quadrata.

PROBLEMA 3

Progettare e codificare in C una procedura che date due frazioni (vedi esempio 2), ne calcoli il prodotto ridotto ai minimi termini.

Procedure – Soluzione 1 (1)

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

void minemax(int, int, int, int*, int*);

void main(void) {
       int x,y,z;
       int MIN, MAX;

       printf("Inserisci il primo intero: ");
       scanf("\n%d",&x);
       printf("Inserisci il secondo intero: ");
       scanf("\n%d",&y);
       printf("Inserisci il terzo intero: ");
       scanf("\n%d",&z);

       minemax(x,y,z, &MIN, &MAX);
       printf("\n\nMinimo: %d", MIN);
       printf("\nMassimo: %d\n", MAX);
}
```

Procedure – Soluzione 1 (2)

```
void minemax(int a, int b, int c, int *min, int *max)
{
	*min = a;
	*max = a;

	if(b > a && b > c) {
		*max = b;
	}
	if(c > b && c > a) {
		*max = c;
	}
	if(b < a && b < c) {
		*min = b;
	}
	if(c < b && c < a) {
		*min = c;
	}

}
```

Procedure – Soluzione 2 (1)

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

int quadperf(int, int*);

void main(void) {
	int num;
	int rad;

	printf("Inserisci un numero: ");
	scanf("\n%d", &num);
	if(quadperf(num, &rad)) {
		printf("\n\nIl numero e' un quadrato");
		printf("\nLa sua radice: %d\n", rad);
	}
	else
		printf("Il numero non e' un quadrato");
}
```

Procedure – Soluzione 2 (2)

```
int quadperf(int N, int *R) {
     double rad_d = sqrt((double)N);
     int rad_i = (int)rad_d;

     if(rad_i == rad_d) {
          *R = rad_i;
          return 1;
     }
     else {
          return 0;
     }
}
```

Procedure – Soluzione 3 (1)

```
#include <stdio.h>

typedef struct{
      int num;
      int den;
} frazione;

int mcd(int a,int b)
{ if (a==b) return a;
  return (a>b) ? mcd(a-b,b) : mcd(b-a,a);
}

void minterm(frazione *f)
{ int m = mcd((*f).num,(*f).den);
  (*f).num = (*f).num / m;
  (*f).den = (*f).den / m;
}
```

Procedure – Soluzione 3 (2)

```
void prod_minterm(frazione, frazione, frazione*);

void main(void) {
  frazione f_1;
  frazione f_2;
  frazione f_prod;

  printf("Inserisci un termine [NUM/DEN]: ");
  scanf("\n%d/%d",&f_1.num, &f_1.den);

  printf("Inserisci il secondo termine [NUM/DEN]: ");
  scanf("\n%d/%d",&f_2.num, &f_2.den);

  prod_minterm(f_1, f_2, &f_prod);

  printf("\n\nIl prodotto ridotto:\n\t\t\t %d\n", f_prod.num);
  printf("\t\t\t---\n");
  printf("\t\t\t %d\n", f_prod.den);
}
```

Procedure – Soluzione 3 (3)

```
void prod_minterm(frazione f1, frazione f2,
frazione *fr) {
     (*fr).num = f1.num*f2.num;
     (*fr).den = f1.den*f2.den;

     minterm(fr);
}
```

PROBLEMA 4

Crypt-decrypt

Scrivere due procedure per criptare e decriptare un testo.

Serve una parola segreta come chiave: ad esempio “abba” che, in numeri, corrisponda a 1 2 2 1

Per esempio, per il messaggio “helloworld” faremo:

	h	e	l	l	o	w	o	r	l	d
+	1	2	2	1	1	2	2	1	1	2
=	i	g	n	m	p	y	q	s	m	f

Ripetiamo la chiave per tutta la lunghezza del messaggio e scaliamo le lettere di un numero corrispondente.

N.B. Nel linguaggio C i caratteri sono visti come interi e vale **‘a’==97** e **‘z’==122**

Procedure – Soluzione 4 (1)

```
#include <stdio.h>
#include <string.h>
#define DIM_STR            101
#define DIM_KEY            16

void encrypt(char*, char*);
void decrypt(char*, char*);
void main(void) {
  char str[DIM_STR];
  char key[DIM_KEY];
  printf("Inserisci una stringa (max %d char):\n", DIM_STR);
  gets(str);
  fflush(stdin);
  printf("\nInserisci la chiave (max %d char):\n", DIM_KEY);
  scanf("%s",key);

  encrypt(str, key);
  printf("\n\nStringa cifrata:\n%s", str);

  decrypt(str, key);
  printf("\n\nStringa decifrata:\n%s\n\n", str);
}
```

Procedure – Soluzione 4 (2)

```
void encrypt(char *p, char *key) {
     int len_p = (int)strlen(p);
     int len_k = (int)strlen(key);
     int i,j;

     j=0;
     for(i=0; i<len_p; i++) {
          p[i] = p[i] + (key[j]-96);
          j++;
          if(j >= len_k)
               j=0;
     }
}
```

Procedure – Soluzione 4 (3)

```
void decrypt(char *c, char *key) {
     int len_p = (int)strlen(c);
     int len_k = (int)strlen(key);
     int i,j;

     j=0;
     for(i=0; i<len_p; i++) {
          c[i] = c[i] - (key[j]-96);
          j++;
          if(j >= len_k)
               j=0;
     }
}
```

Procedure – Soluzione 5

Esercizio 5

- Uno studente deve calcolare la media dei suoi esami per la laurea. Gli esami non sono tutti uguali: a ciascun esame è associato un numero di crediti. La media degli esami è la media pesata, considerando i crediti come pesi.

- Si scriva un programma che
  - definisce un tipo di dato “esame” che contiene sia il numero di crediti sia il voto dell’esame
  - legge da tastiera i dati sugli esami sostenuti da uno studente (crediti e voto per ogni esame)
  - calcola la media pesata

Procedure – Soluzione 5 (1)

```
#include <stdio.h>
#define MAX 20

typedef struct {
      char nome[10];
      float voto;
      int crediti;
} esame;

float mediaesami(esame e[], int n) {
      float temp = 0;
      int i, creditititotali = 0;
      for(i=0; i<n; i++) {
            temp += e[i].voto * e[i].crediti;
            creditititotali += e[i].crediti;
      }
      return temp/creditititotali;
}
……
```

Procedure – Soluzione 5 (2)

```
void main() {
  int numeroesami, i;
  float media;
  esame esami[MAX];

  printf("Numero di esami: ");
  scanf("\n%d", &numeroesami);

  for(i=0; i<numeroesami; i++) {
    printf("Nome esame: "); scanf("\n%s", esami[i].nome);
    printf("Crediti: "); scanf("\n%d", &esami[i].crediti);
    printf("Voto: "); scanf("\n%f", &esami[i].voto);
    printf("\n");
  }

  media = mediaesami(esami, numeroesami);
  printf("La media pesata degli esami e': %.2f\n", media);
}
```