

ESAME PROGRAMMAZIONE II



Alessio Piromallo

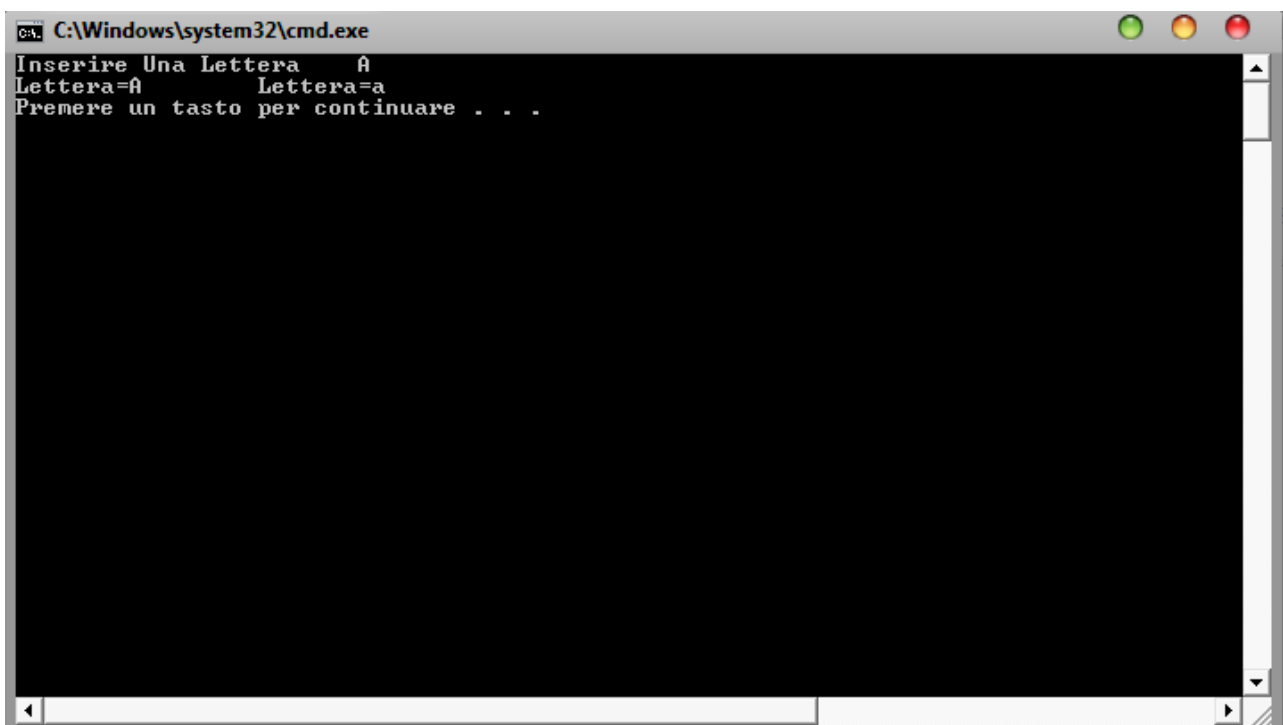
0108001016

P2_01_01_AC

1. [liv.1] Scrivere una function C che cambia il carattere in input da minuscolo a maiuscolo e viceversa automaticamente.

```
#include <stdio.h>

int main() {
    char p1='A';
    int swich =32;
    char p2='B';
    printf("Inserire Una Lettera  ");
    scanf("%c",&p1);
        // Swiccia il valore inserito P1 di 32 bit 2^5
        // 00100000
    p2=(p1^swich);
    printf("Lettera=%c\t Lettera=%c\n",p1,p2);
    getchar();
    return (0);
}
```

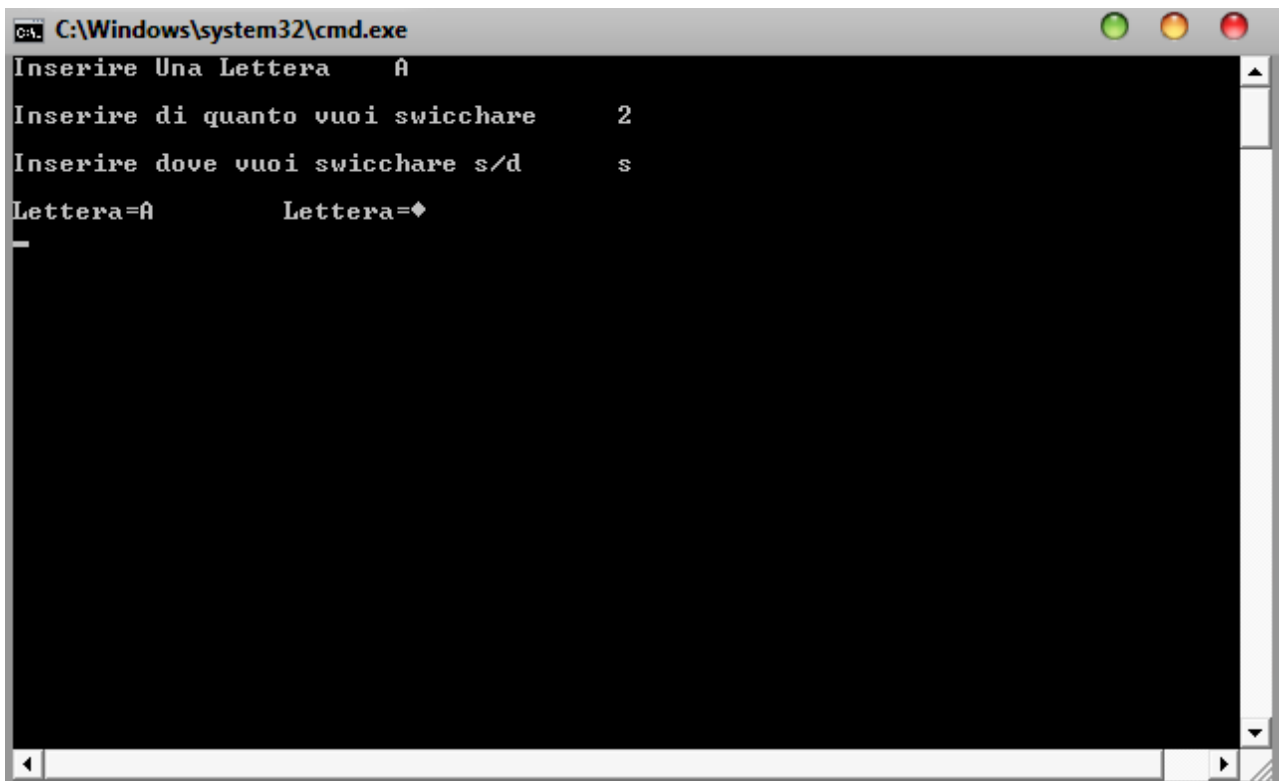


```
C:\Windows\system32\cmd.exe
Inserire Una Lettera  A
Lettera=A      Lettera=a
Premere un tasto per continuare . . .
```

2.[liv.1] Scrivere una function C per ruotare di n bit, verso sinistra o verso destra (stabilito in input), il contenuto di una variabile mediante gli operatori bitwise.

```
#include <stdio.h>
int main() {
    char p1='x', p2='y', lswich='ò';
    int swich=1;

    printf("Inserire Una Lettera  ");
    scanf("%c",&p1);
    getchar();
    printf("\nInserire di quanto vuoi swicchare  ");
    scanf("%d",&swich);
    getchar();
    do{
        printf("\nInserire dove vuoi swicchare s/d  ");
        scanf("%c",&lswich);
        getchar();
        if(lswich=='s'){
            //Swiccia a Sinistra di N bit
            p2=(p1<<swich);}
        if(lswich=='d'){
            //Swiccia a Destra di N bit
            p2=(p1>>swich);}}
    while ((lswich!='s' && lswich!='d'));
    printf("\nLettera=%c\t Lettera=%c\n",p1,p2);
    getchar();
    return(0);}
```



```
C:\Windows\system32\cmd.exe
Inserire Una Lettera  A
Inserire di quanto vuoi swicchare  2
Inserire dove vuoi swicchare s/d  s
Lettera=A          Lettera=◆
```

P2_01_02_AC

1.[liv.1] Scrivere una function *C* che, dopo aver estratto i bit da una variabile intera *X*, ne calcola il relativo valore dalla formula:

$$\text{Val_X} = b_{n-1}2^{n-1} + \dots + b_{22}2^2 + b_{21}2^1 + b_{20}2^0$$

dove *b* è l'array dei bit di *X*. Confrontare il risultato con il valore della variabile *X* dichiarata una volta signed ed un'altra unsigned.

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define max_len 32
void BW(short ,char [], short []);
int main()
{
    short vm=0; //scelta ne menu
    short bit[max_len],d=0;
    union word32bit {long L;
                     char C[4];
                     short S[2];
                    } word;

    do{
        printf("Seleziona "); printf("0 uscire dal programma\n");
        printf("1 Per gestire un valore Long\t");
        printf("2 Per gestire un valore Char\t");
        printf("3 Per gestire un valore Short\t");
        scanf("%d",&vm);
        fflush(stdin);
        switch(vm){ // Menu valori da controllare
            case 1: printf("\nInmettere un valore long\t");
                    scanf("%d",&(word.L));
                    fflush(stdin);
                    printf("\nBINARIO ");
                    d=sizeof(long);
                    BW(d,word.C,bit);
                    printf("");break;

            case 2: printf("\nInmettere un valore char\t");
                    scanf("%c",&(word.C[0]));
                    fflush(stdin);
                    printf("\nBINARIO ");
                    d=sizeof(char);
                    BW(d,word.C,bit);
                    printf("");break;

            case 3: printf("\nInmettere un valore short\t");
                    scanf("%d",&(word.S[0]));
                    fflush(stdin);
                    printf("\nBINARIO ");
                    d=sizeof(short);
                    BW(d,word.C,bit);
                    printf("");break;

            default : break;
        }
    } while(vm!=0);

    return (0);
```

```

}

//-----

void BW(short len,char ch[], short bit[max_len])
{
    short j,jc;
    char c;
    signed int calc=0;

    for(j=0; j<max_len;j++)
        bit[j]=0; // array imp a 0
    for(jc=0; jc<len; jc++)
        {c=ch[jc]; //avanza in ordine dei bayt
        for(j=0; j<8; j++)
            {bit[j+8*jc]=c&1;
            c=c>>1;
            }
        }

    // visualizza i bit
    for(j=8*len-1; j>=0; j--)
    {printf("%d",bit[j]);
    if(!(j%4))
        {printf(" ");}
    if(bit[j]==1){calc=calc+pow(2.0,j);}
    }
    printf("Decimale %d",calc);
    printf("\n");
}

```

```

C:\Windows\system32\cmd.exe
Selezione 0 uscire dal programma
1 Per gestire un valore Long      2 Per gestire un valore Char      3 Per gestire
valore Short      1

Inmettere un valore long          3

BINARIO 0000 0000 0000 0000 0000 0000 0000 0011 Decimale 3
Selezione 0 uscire dal programma
1 Per gestire un valore Long      2 Per gestire un valore Char      3 Per gestire
valore Short      2

Inmettere un valore char          D

BINARIO 0100 0100 Decimale 68
Selezione 0 uscire dal programma
1 Per gestire un valore Long      2 Per gestire un valore Char      3 Per gestire
valore Short      -

```

2. *[liv.1] Scrivere una function C per estrarre dalla variabile intera X i k bit più significativi o meno significativi, usando:*

1) *Una maschera.*

2) *Gli operatori di shift.*

3) *Prodotto o divisione per potenze di 2.*

```
#include<stdio.h>
#include<math.h>
#define lan 8
void bit(short ,short a[]);
int maschera(short A, short y, char sig,short scelta);

int main(){
    char sig='A',exit='D';
    short A=0;
    short scelta=0;
    short y=0;
    short decimale=0;
    short a[lan];
    do{
        printf("Inserisci un intero\t");
        scanf("%d",&A);
        getchar();
        bit(A,a);
        printf("\nPrendi in cons. i valori piu sig o meno sig ? S/D\t");
        scanf("%c",&sig);
        getchar();
        printf("\nQuanti bit vuoi prendere in considerazione ?\t");
        scanf("%d",&y);
        getchar();
        printf("1 per usare una maschera \t");
        printf("\n2 per usare unlo shift >> o <<\t");
        printf("\n3 per usare prodotti o divisione di potenze\n");
        scanf("%d",&scelta);
        getchar();
        decimale=maschera(A,y,sig,scelta);
        A=decimale;
        bit(A,a);
        printf("\n");
        printf("Vuoi uscire S/N\t");
        scanf("%c",&exit);
        getchar();
    } while(exit=='N');
    printf("\n");
    return(0);
    getchar();
}
```

```

int maschera(short A, short y, char sig,short scelta)
{
    short n=0,swich=0, mask=0, b=0;
    double mask_s=0.0,mask_d=0.0,j=0.0;

    // crea una maschera sfruttando il for
    // successivamente avviene una & tra la maschera e il valore inserito
    if(scelta==1){
        for(b=0; b<y; b++)
        {mask=mask<<1|1;}

        if (sig=='D'){
            mask=(A & mask);
            printf("A=%d bit da swiccare=%d piu Sig sono formati dai valori decimali %d\n",A,y,mask);
        }

        if (sig=='S'){
            mask=mask<<(lan-y);
            mask=mask&A;
            printf("A=%d bit da swiccare=%d meno Sig sono formati dai valori decimali %d\n",A,y,mask);
        }
    }

    //inserisce nel valore mask il risultato del valore A shiftato di "y" bit
    if(scelta==2){
        if (sig=='S'){
            mask=A<<y;
            printf("L'intero A %d swicciato di %d bit ha come numero decimale piu significativo %d\n",A,y,mask);
        }
        if (sig=='D'){
            mask=A>>y;
            printf("L'intero A %d swicciato di %d bit ha come numero decimale piu significativo %d\n",A,y,mask);
        }
    }

    //crea una maschera tramite pow e successivamente
    //una & bit a bit tra la maschera e il valore intero
    if(scelta==3)
    {
        if(sig=='S')
        {
            for(j=lan-1; j>=(lan-y); j--)
            {
                mask_d=mask_d+pow(2.0,j);
            }
            mask=((short)mask_d)&A;
            printf("Il numero decimale A %d al corrispettivo numero di bit piu significativi e %d\n",A,mask);
        }
        if(sig=='D')
        {
            for(j=0; j<y; j++)
            {
                mask_s=mask_s+pow(2.0,j);
            }
            mask=((short)mask_s)&A;
            printf("Il numero decimale A %d al corrispettivo numero di bit meno significativi e %d\n",A,mask);
        }
    }
}

```

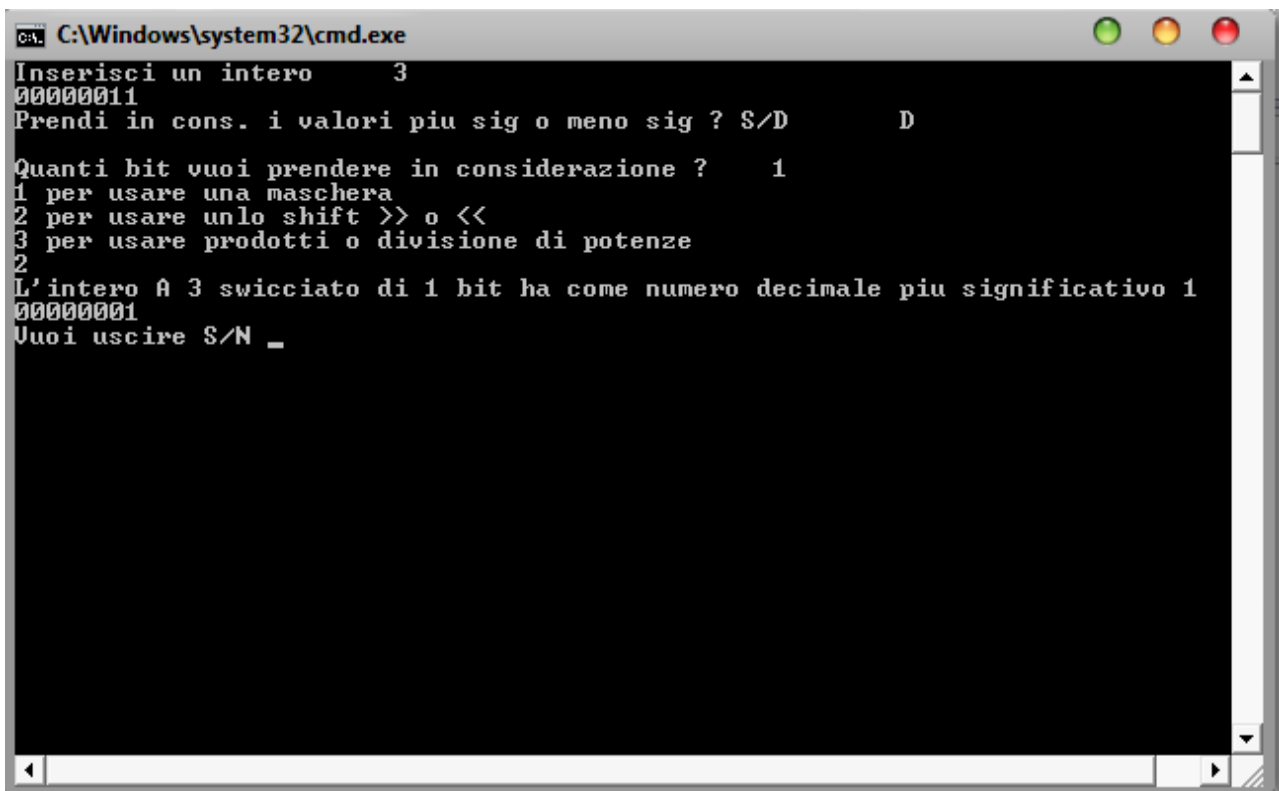
```

return(mask);
}

// trasforma in numero da modulo 10 a 2
void bit(short A,short a[lan])
{
    short j=(lan-1);
    do{
        a[j]=A&1; --j;
        A=A>>1;
    }while (A!=0 && j>=0);
    if(j>0){
        do{
            a[j]=0; --j;
        }while (j>=0);}

    for(j=0; j<lan; j++)
        printf("%d", a[j]);
}

```



```

C:\Windows\system32\cmd.exe
Inserisci un intero      3
00000011
Prendi in cons. i valori piu sig o meno sig ? S/D      D

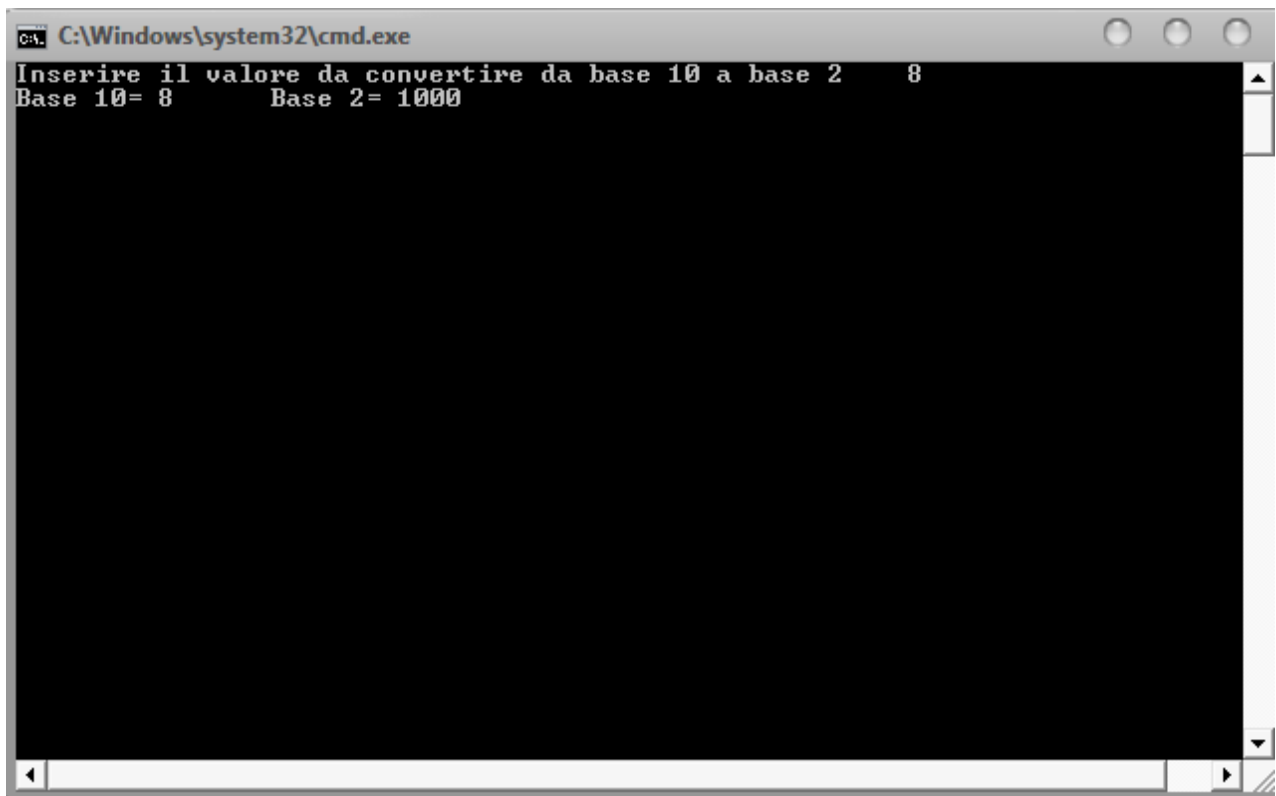
Quanti bit vuoi prendere in considerazione ?      1
1 per usare una maschera
2 per usare unlo shift >> o <<
3 per usare prodotti o divisione di potenze
2
L'intero A 3 swicciato di 1 bit ha come numero decimale piu significativo 1
00000001
Vuoi uscire S/N _

```


P2_02_01_AT

1.[liv.1] Scrivere una function C di conversione di un intero positivo da base 10 a base 2 mediante l'algoritmo delle divisioni successive.

```
#include <stdio.h>
#define Max 100
void conversione(short, short R[]);
int main(){
    short R[Max];
    short A=0;
    printf("Inserire il valore da convertire da base 10 a base 2\t");
    scanf("%d",&A);
    getchar();
    conversione(A,R);
    return(0);
    getchar();
}
void conversione(short A, short R[]){
    short j=0,Q1=0,Q=A;
    printf("Base 10= %d    Base 2= ",A);
    while(Q>0){
        j=j+1;
        //divide il valore per 1\2
        Q1=(Q/2);
        // calcola il rispettivo valore bit a bit
        R[j]=Q-(Q1*2);
        Q=Q1;
    }
    for (j=j;j>0;j--){
        printf("%d",R[j]);
    }
    printf("\n");
}
```



2.[liv.1] Scrivere una function C di conversione di un intero positivo da base 2 a base 10 che generi un array di caratteri contenenti le cifre decimali.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define MAX 1001
void conversione(short B[],short D[]);

//B[MAX] Array Binario vengono salvati i bit
//D[MAX] Array Decimale vengono salvati i valori in decimale dei bit di B[MAX]
int main(){
short B[MAX],D[MAX];
conversione(B,D);
return(0);
}

void conversione(short B[MAX],short D[MAX]){
short somm_D=0,exp=0,max_j=0,y=0,j=-1;
printf("Inserire i corrispondente in base 2 singolarmente\t");
do{ //Inserisco i valori singolarmente nell array
j=j+1;
scanf("%d",&B[j]);
}while (B[j]==0 || B[j]==1);
max_j=j;
exp=max_j-1; //esponente per la conversione di ogni bit
for(j=0; j<max_j; j++)
{
D[y]=(pow(2.0,exp)*B[j]);// convesione ogni singolo bit in decimale pow(2,y)
y=(y+1);
exp=exp-1;
}
system("CLS");
printf("Array binario= ");
```

```

for(j=0;j<max_j;j++)
{ printf("%d",B[j]); }
printf(" Array decimale= ");
for(y=0;y<max_y;y++)
{
    somm_D=somm_D+D[y];
    printf("%d ",D[y]);
}

printf("=%d",somm_D);
printf("\n\n");
}

```

```

C:\Windows\system32\cmd.exe
Array binario= 00110 Array decimale= 0 0 4 2 0 =6
Premere un tasto per continuare . . . _

```

3.[liv.2] Ripetere l'esercizio precedente nel caso che l'input sia una stringa di caratteri contenenti i bit del numero.

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define MAX 1001
void conversione(char B[],short D[]);

int main(){
//B[MAX] Array Binario vengono salvati i bit
//D[MAX] Array Decimale vengono salvati i valori in decimale dei bit di B[MAX]
short D[MAX];
char B[MAX];
conversione(B,D);
return(0);
}

void conversione(char B[MAX],short D[MAX]){
short somm_D=0,exp=0,max_j=0,y=0,j=-1,max_y=0;
printf("Inserire i corrispondente in base 2 singolarmente\t");
do{ //Inserisco i valori singolarmente nell array
    j=j+1;
    scanf("%c",&B[j]);
    getchar();
}while (B[j]!='0' || B[j]!='1');

max_j=j;
exp=max_j-1; //esponente per la conversione di ogni bit
for(j=0; j<max_j; j++)
{
    if(B[j]=='1')

```

```

        {
            D[y]=(pow(2.0,exp)); // conversione ogni singolo bit in decimale pow(2,y)
            y=(y+1);
            exp=exp-1;
        }
        if(B[j]=='0')
        {
            D[y]=0; // conversione ogni singolo bit in decimale (0*pow(2,y)) da 0
            y=(y+1);
        }
    }
    max_y=y;
    system("CLS");
    printf("Array binario= ");
    for(j=0;j<max_y;j++)
    {
        printf("%c",B[j]);
    }
    printf(" Array decimale= ");
    for(y=0;y<max_y;y++)
    {
        somm_D=somm_D+D[y];
        printf("%d ",D[y]);
    }
    printf("=%d",somm_D);
    printf("\n\n");
}

```

```

C:\Windows\system32\cmd.exe
Array binario= 1100 Array decimale= 8 4 0 0 =12
Premere un tasto per continuare . . . _

```

P2_02_02_AT

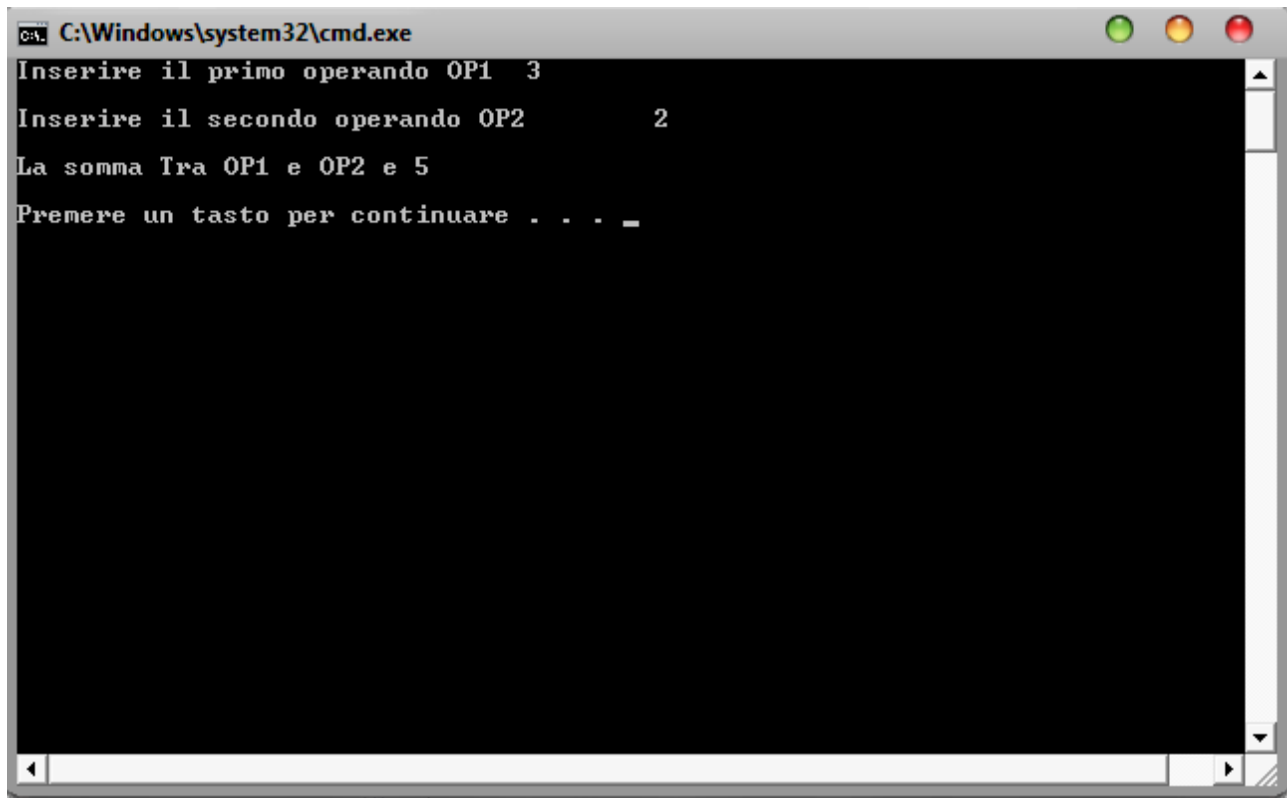
1.[liv.1] Scrivere una function C per l'addizione aritmetica binaria mediante gli operatori bitwise.

```
#include <stdio.h>

void Addizione(int op1,int op2);

int main(){
int op1=0,op2=0;
printf("Inserire il primo operando OP1\t");
scanf("%d",&op1);
getchar();
printf("\nInserire il secondo operando OP2\t");
scanf("%d",&op2);
getchar();
Addizione(op1,op2);
getchar();
return(0);}

void Addizione(int op1,int op2){
    int sum=0,rest=1;
    while(rest>0){
        sum=op1^op2;
        rest=op1&op2; rest=rest<<1;
        op1=sum; op2=rest;
    }
    printf("\nLa somma Tra OP1 e OP2 e %d\n",op1);
}
```



```
C:\Windows\system32\cmd.exe
Inserire il primo operando OP1 3
Inserire il secondo operando OP2 2
La somma Tra OP1 e OP2 e 5
Premere un tasto per continuare . . . _
```

2.[liv.1] Scrivere una function C per la sottrazione aritmetica binaria in base 2 di due numeri naturali (N) mediante gli operatori bitwise.

```
#include <stdio.h>
void Addizione(int op1,int op2);

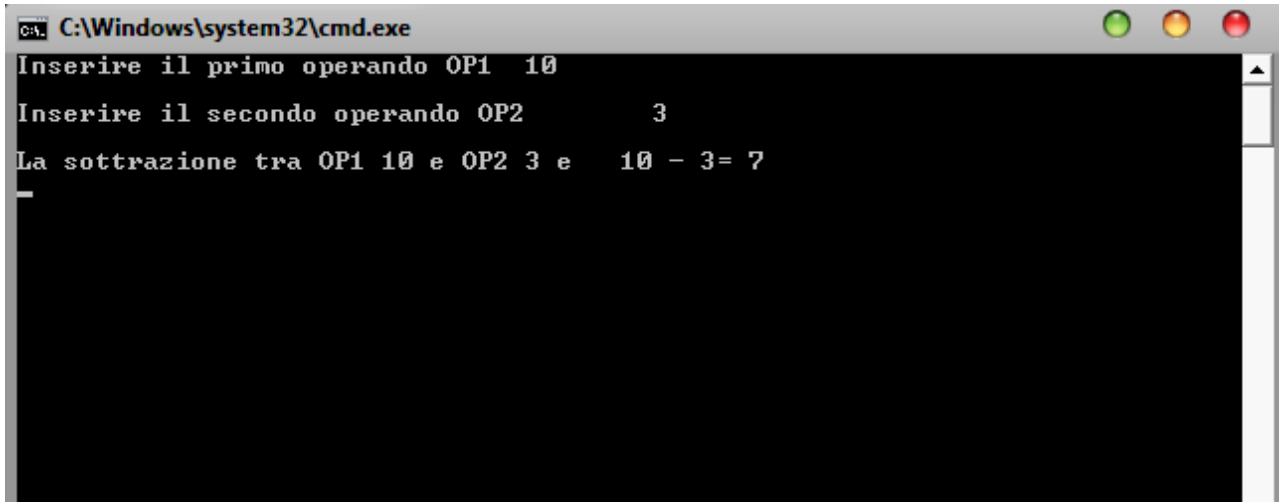
int main(){
    int op1=0,op2=0;
    printf("Inserire il primo operando OP1\t");
    scanf("%d",&op1);
    getchar();
    printf("\nInserire il secondo operando OP2\t");
    scanf("%d",&op2);
    getchar();
    Addizione(op1,op2);
    getchar();
    return(0);
}

void Addizione(int op1,int op2){
    int sum=0,rest=1;
    if(op2>op1){//op2>op1 si scambiano
        op1=op1^op2;
        op2=op1^op2;
        op1=op1^op2;
    }
    printf("\nLa sottrazione tra OP1 %d e OP2 %d e  %d - %d= ",op1,op2,op1,op2);
    while(rest>0){
        sum=op1^op2;
        rest= ~(op1|(~op2));//formula prestito per la sottrazione
        rest=rest<<1;
    }
}
```

```

        op1=sum; op2=rest;
    }
    printf("%d\n",op1);
}

```



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The prompt displays the following text: "Inserire il primo operando OP1 10", "Inserire il secondo operando OP2 3", and "La sottrazione tra OP1 10 e OP2 3 e 10 - 3 = 7". A horizontal line is drawn under the result "7".

P2_02_04_AT

1. [liv.1] Scrivere una function C che, fissato il numero n di bit, calcoli la rappresentazione di un intero:

- .1 per complemento a 2 ;
- .2 eccesso B (B -biased).

```

#include <stdio.h>
#include <math.h>
int comp_2(int k);
int biased(int k);

int main()
{
    int i,k,result1,result;
    for(i=-127; i<128; i++)
    {
        result=comp_2(i);
        result1=biased(i);
        printf("Valore %d base2 %d Biased %d\n",i,result,result1);
    }
    return(0);
}

// Nega Bit a Bit il valore di K e aggiunge 1 = 0001
int comp_2(int k)
{ return(~k)+1; }

// Calcola il valore del biased
int biased(int k)
{
    int c,a;

```

```
// C prende il valore in bait di k e lo trasforma in bit sottraendo 0001 per il segno
    c=(8*sizeof(k)-1);
    return (c-k)+1;
}
```

```
Valore -127 base2 127 Biased 159
Valore -126 base2 126 Biased 158
Valore -125 base2 125 Biased 157
Valore -124 base2 124 Biased 156
Valore -123 base2 123 Biased 155
Valore -122 base2 122 Biased 154
Valore -121 base2 121 Biased 153
Valore -120 base2 120 Biased 152
Valore -119 base2 119 Biased 151
Valore -118 base2 118 Biased 150
Valore -117 base2 117 Biased 149
Valore -116 base2 116 Biased 148
Valore -115 base2 115 Biased 147
Valore -114 base2 114 Biased 146
Valore -113 base2 113 Biased 145
Valore -112 base2 112 Biased 144
Valore -111 base2 111 Biased 143
Valore -110 base2 110 Biased 142
Valore -109 base2 109 Biased 141
Valore -108 base2 108 Biased 140
Valore -107 base2 107 Biased 139
Valore -106 base2 106 Biased 138
Valore -105 base2 105 Biased 137
Valore -104 base2 104 Biased 136
Valore -103 base2 103 Biased 135
Valore -102 base2 102 Biased 134
Valore -101 base2 101 Biased 133
Valore -100 base2 100 Biased 132
Valore -99 base2 99 Biased 131
Valore -98 base2 98 Biased 130
Valore -97 base2 97 Biased 129
Valore -96 base2 96 Biased 128
Valore -95 base2 95 Biased 127
Valore -94 base2 94 Biased 126
Valore -93 base2 93 Biased 125
Valore -92 base2 92 Biased 124
Valore -91 base2 91 Biased 123
Valore -90 base2 90 Biased 122
Valore -89 base2 89 Biased 121
Valore -88 base2 88 Biased 120
Valore -87 base2 87 Biased 119
Valore -86 base2 86 Biased 118
Valore -85 base2 85 Biased 117
Valore -84 base2 84 Biased 116
Valore -83 base2 83 Biased 115
Valore -82 base2 82 Biased 114
Valore -81 base2 81 Biased 113
Valore -80 base2 80 Biased 112
Valore -79 base2 79 Biased 111
Valore -78 base2 78 Biased 110
Valore -77 base2 77 Biased 109
Valore -76 base2 76 Biased 108
Valore -75 base2 75 Biased 107
Valore -74 base2 74 Biased 106
Valore -73 base2 73 Biased 105
Valore -72 base2 72 Biased 104
Valore -71 base2 71 Biased 103
Valore -70 base2 70 Biased 102
```


2.[liv.1] Scrivere una function C per l'addizione algebrica binaria di due interi (\square) mediante gli operatori bitwise, traducendo l'algoritmo di seguito riportato:

```
{Algoritmo di "addizione binaria" mediante operatori sui bit}
    rip:=1;
    while rip>0
        sum:=bitXOR(op1,op2);
        rip:=bitAND(op1,op2);
        rip:=leftSHIFT(rip,1);
        op1:=sum; op2:=rip;
    endwhile
```

Se l'operazione da implementare deve essere l'addizione algebrica (cioè deve valere anche per gli interi negativi rappresentati per complemento a 2), quale accorgimento va usato nella traduzione in C dell'algoritmo ... e perché.

```
#include <stdio.h>
#include <math.h>
short addo(short op1, short op2);
void bit_show(short a, short b[]);
int main()
{
    short a=0, op1, op2, somma, b[100];
    char c;
    do{
        printf("Inserire il primo Operando\t");
        scanf("%d",&op1);
        fflush(stdin);
        printf("\nInserire il secondo Operando\t");
        scanf("%d",&op2);
        fflush(stdin);
        printf("\n");
        bit_show(op1,b);
        printf("\n");
        bit_show(op2,b);
        printf("\n");
        printf("-----\n");
        somma=addo(op1,op2);
        bit_show(somma,b);
        printf("\n\nLa somma tra %d + %d = %d\t",op1,op2,somma);
        bit_show(somma,b);
        printf("\n");
        printf("Esci S/N\t");
        scanf("%c",&c);
    } while(c=='N' || c=='n');
    printf("\n\n\n");
    return 0;
}
```

/* permette di fare la somma binaria solo sui 4 bit meno sig così da permettere di conservare il segno in caso sia un numero negativo o positivo */

```
short addo(short op1, short op2)
{
    short j, rip, som, op1_f=(op1), op2_f=(op2);
    rip=1;
    som=0;
    for(j=0;j<=14;j++)
    {
        som=(op1_f^op2_f);
        rip=(op1_f&op2_f);
        rip=(rip<<1);
        op1_f=som;
        op2_f=rip;
    }
    som=op1_f;
    return som;
}
```

/* Visualizza tutti i bit che compongono il numero decimale */

```
void bit_show(short a, short b[])
{
    short j;
    for(j=0;j<16;j++)
    {
        b[j]=a&1;
        a=a>>1;
    }

    for(j=16-1;j>=0;j--)
    {
        printf("%d", b[j]);
        if(!(j%4)){printf(" ");}
    }
}
```

}

```
C:\Windows\system32\cmd.exe
Inserire il primo Operando      -2
Inserire il secondo Operando    3

1111 1111 1111 1110
0000 0000 0000 0011
-----
0000 0000 0000 0001

La somma tra -2 + 3 = 1 0000 0000 0000 0001
Esci S/N          s

Premere un tasto per continuare . . .
```

P2_03_02_AT

1.[liv.1] Scrivere una function C per visualizzare la rappresentazione binaria (s,e,m) di un numero float. Verificare che il valore del numero ottenuto coincida con il dato iniziale.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
void estrarre_bit(long ,short []);
float conversione(short [],float);
int main(void)
{
    char i,sce;
    float contr=0.0;
    short bit[32];

    //il valore inserito in a.fa occupa la
    //stessa allocazione di memoria di a.la

    union base_singola { long la; float fa; }a;
    do{
        system("cls");
        printf("Inserire il numero FLOAT da trasformare\t");
        scanf("%f",&(a.fa));
        getchar();
        estrarre_bit(a.la,bit);
        system("cls");
        printf("Float=%f\t",a.fa);
        printf("Esp=%e\t",a.fa);
        printf("Hex=%08x\n",a.fa);
        printf("Bit Corrispondenti\t Segno  Esponente  Mantissa\n");
        printf("\t\t\t\t %1d\t\t\t\t",bit[0]);
        for(i=1; i<=8; i++)
            printf("%1d",bit[i]);
```

```

printf(" ");
for(i=9; i<32; i++)
    printf("%1d",bit[i]);

contr=conversione(bit,contr);
if(a.fa==contr)
{
    printf("\nYES! X=%f e uguale a %f",a.fa,contr);
}
if(a.fa!=contr)
{
    printf("\nNO! X=%f non e uguale a %f",a.fa,contr);
}

printf("\n\n");
printf("Vuoi ripetere il ProgrammaS/N\t");
scanf("%c",&sce);
getchar();
} while(sce=='S' || sce=='s');
return 0;
}

```

//Carica ogni singolo bit nell array bit
// con una AND tra il valore 1 reg(a.la)
// successivamente shiftato di 1 a >>

```

void estrarre_bit(long reg,short bit[32])
{
    long uno=1; short i;
    for(i=31;i>=0;i--)
    {
        bit[i]=(reg&uno);
        reg=reg>>1;
    }
}

```

//segundo la funzione $2^{(e-127)} * (m/8388608)$
//faccio scorrere tutti i valori dell array bit e
//prendo in considerazione la mantissa con valori ==1
//addizionandola se stessa a $pow(2,31-j)$

```

float conversione(short bit[32],float contr)
{
    int j,i,e=0,x=0;
    float n=8388608.0,m=0.0;
    printf("\n");
    for(j=31;j>=9;j--)
    {
        if(bit[j]==1)
        {
            if(j==31){x=0;}
            if(j<31){x=(31-j);}
            m=pow(2.0,x)+m;
        }
    }
    x=0;
}

```

//segundo la funzione $2^{(e-127)} * (m/8388608)$
//faccio scorrere tutti i valori dell array bit e

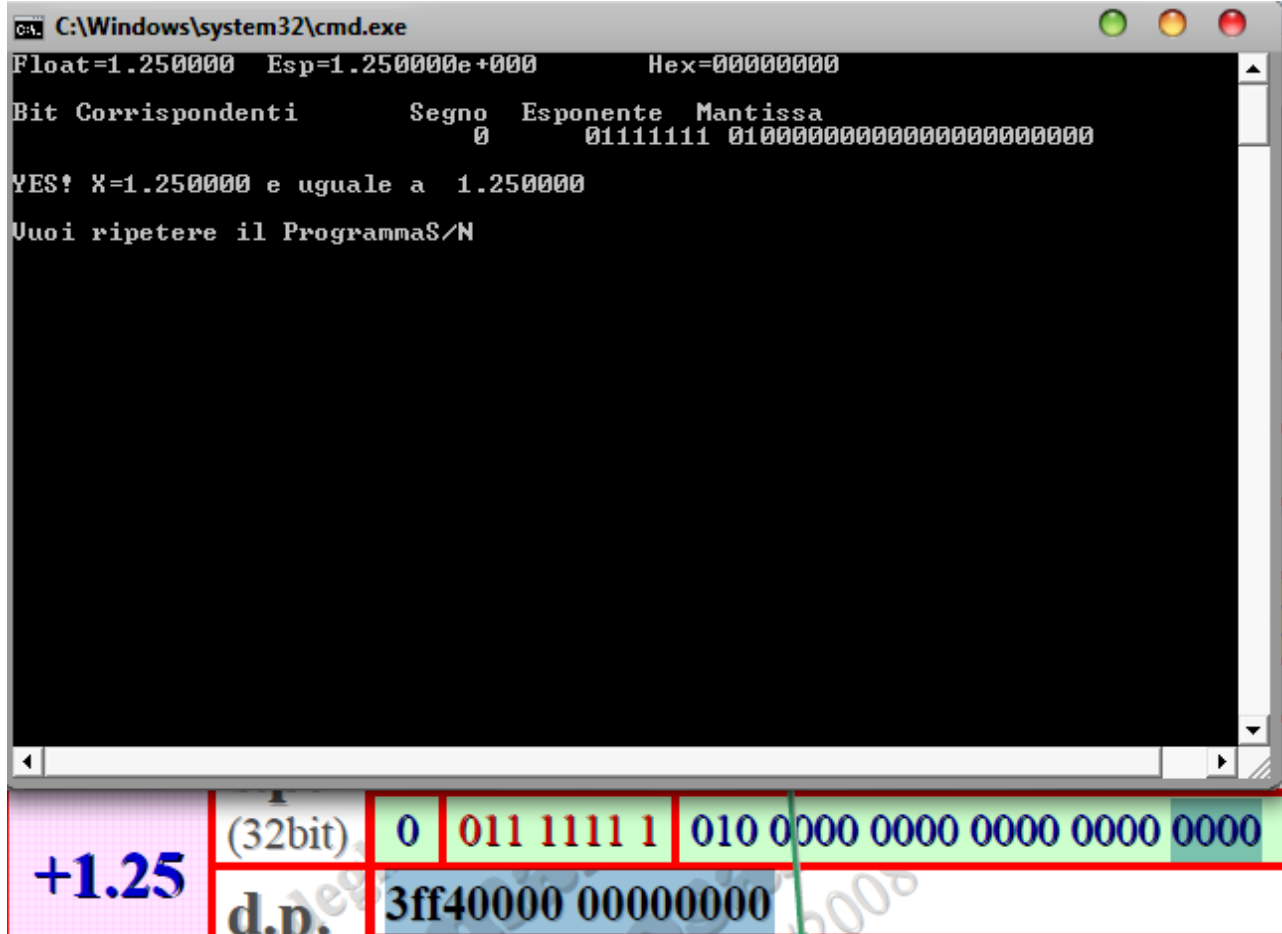
```

//prendo in considerazione l'esponente con valori ==1
//addizionandola se stessa a pow(2,8-i)

    for(i=8;i>=1;i--)
    {
        if(bit[i]==1)
        {
            if(i==8){x=0;}
            if(i<8){x=(8-i);}
            e=pow(2.0,x)+e;
        }
    }
    e=e-127;
//eseguo la funzione  $2^{(e-127)} \cdot (m/8388608)$ 

    return(contr=(1+(m/n))*(pow(2.0,e)));
}

```



P2_03_03_AT

1.[liv.2] Generando in modo random i bit [vedi pdf delle dispense] di un numero reale x (double x), determinare i bit della corrispondente rappresentazione float flx (float flx ; $flx = (\text{float}) x$). Se il numero x è rappresentabile nel tipo float, calcolarne l'errore assoluto EA e l'errore relativo ER (considerando come esatto double x e come approssimante float flx) dalle formule:

$$EA(fl x) = |x - fl x| \quad ER(fl x) = \frac{|x - fl x|}{|x|}$$

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#define RAND_X 32767
void estrarre_bit(short bit[64]);
double conversione(short bit[64],double contr);
double val_ass(double contr, float contr_f);
double val_rell(double contr, float contr_f);
int main(void)
{
    char c;
    short bit[64];
    double contr=0.0,v_ass=0.0,v_rel=0.0;
    float contr_f=0.0;
    do
    {
        estrarre_bit(bit);
        contr=conversione(bit,contr);
        contr_f=(float)contr;
        v_ass=val_ass(contr,contr_f);
        v_rel=val_rell(contr,contr_f);
        printf("Double= %22.16e\t    Float= %22.16e\n",contr,contr_f);
        printf("Valore ASS.= %22.16e\t    Valore REL.= %22.16e\n",v_ass,v_rel);
        printf("\nVuoi ripetere il programma S/N\t");
        scanf("%c",&c);
        getchar();
        system("cls");
    }while(c=='s' ||c=='S');
return 0;}

```

/* Crea un double(array[64]) con valori 1 o 0
seguendo l' istruzione nella if*/

```

void estrarre_bit(short bit[64])
{
    short i;
    srand((int)(unsigned)time(NULL));
    for(i=0;i<64;i++)
    {
        bit[i]=rand()%RAND_X;
        if(bit[i]<16384)
        {
            bit[i]=0;
        }
        else
        {
            bit[i]=1;
        }
    }
}

// Converte l'array bit in numero decimale
double conversione(short bit[64],double contr)
{
    int j,i,e=0,x=0,m=0;
    double n=4503599627370496.0;
    for(j=63;j>=12;j--)
    {
        if(bit[j]==1)
        {
            if(j==63){x=0;}
            if(j<63){x=(63-j);}
            m=(short)pow(2.0,x)+m;
        }
    }
    x=0;

```

//seguendo la funzione $2^{(e-1023)} * (m/4503599627370496.0)$

//faccio scorrere tutti i valori dell array bit e

```

//prendo in considerazione l'esponente con valori ==1
//addizionandola se stessa a pow(2,11-i)
    for(i=11;i>=1;i--)
    {
        if(bit[i]==1)
        {
            if(i==11){x=0;}
            if(i<11){x=(11-i);}
            e=(short)pow(2.0,x)+e;
        }
    }
    e=e-1023;
//eseguo la funzione  $2^{(e-1023)} \cdot (m/4503599627370496.0)$ 
    contr=(1+(m/n))*(pow(2.0,e));
    return contr;
}
/* calcola l'errore ASS con la seguente funzione
fl(x)=(|x-X|) */
double val_ass(double contr, float contr_f)
{
    double ass=0.0;
    ass=fabs(contr-contr_f);
    return ass;
}
/* calcola l'errore rel con la seguente funzione
fl(x)=(|x-X|)/x */
double val_rell(double contr, float contr_f)
{
    double rel=0.0;
    rel=fabs((contr-contr_f))/fabs(contr);
    return rel;
}

```



```
ca. C:\Windows\system32\cmd.exe
Double= 4.3322963970411411e+127      Float= 1.#INF0000000000000e+000
Valore ASS.= 1.#INF000000000000e+000      Valore REL.= 1.#INF0000000000
+000

Vuoi ripetere il programma S/N s

Double= 3.1352853188247892e+203      Float= 1.#INF0000000000000e+000
Valore ASS.= 1.#INF000000000000e+000      Valore REL.= 1.#INF0000000000
+000

Vuoi ripetere il programma S/N s

Double= 3.1352853188247892e+203      Float= 1.#INF0000000000000e+000
Valore ASS.= 1.#INF000000000000e+000      Valore REL.= 1.#INF0000000000
+000

Vuoi ripetere il programma S/N s

Double= 1.8051943758701194e-276      Float= 0.000000000000000e+000
Valore ASS.= 1.8051943758701194e-276      Valore REL.= 1.000000000000000
+000

Vuoi ripetere il programma S/N s

Double= 1.0947644252506265e-047      Float= 0.000000000000000e+000
Valore ASS.= 1.0947644252506265e-047      Valore REL.= 1.000000000000000
+000

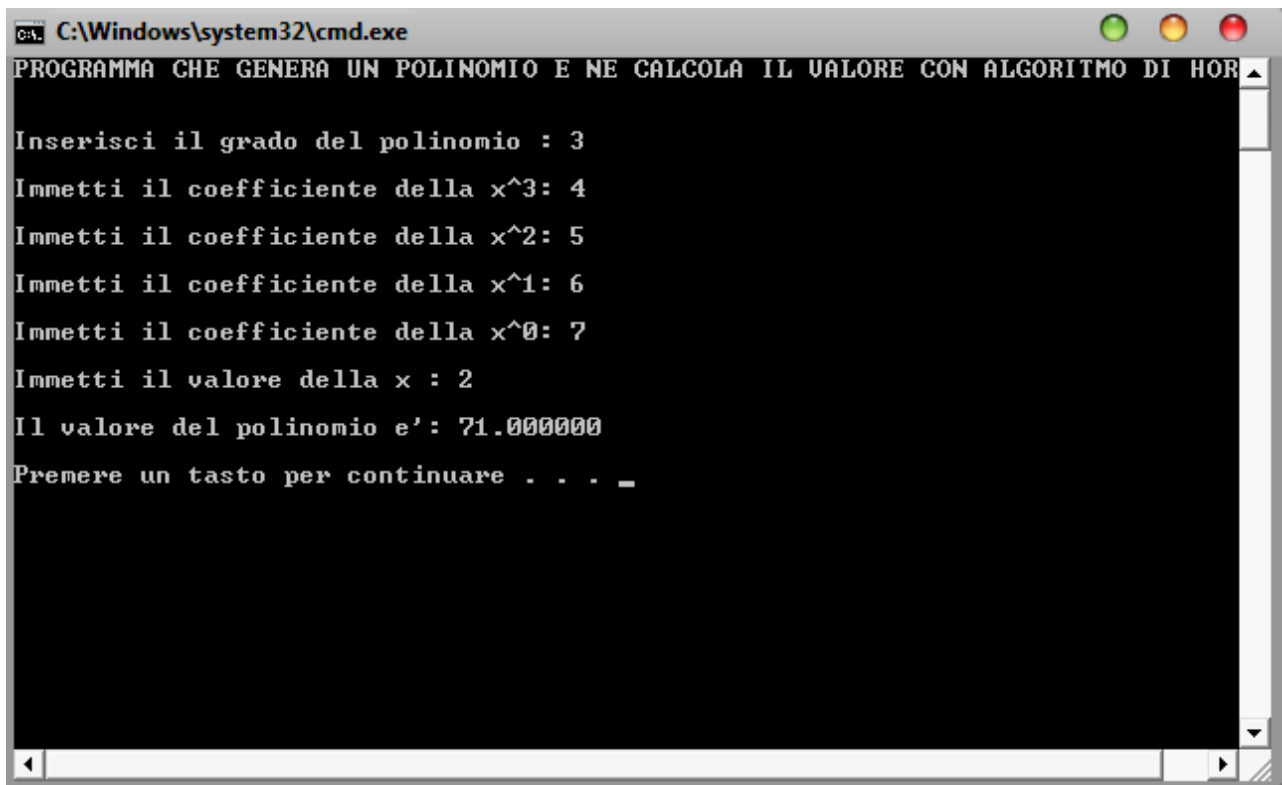
Vuoi ripetere il programma S/N
```

P2_03_04_AT

1.[liv.2] Scrivere una function C per valutare un polinomio mediante algoritmo di Horner.

```
#include <stdio.h>
float horner(int gr, float coef[], float x);
int main()
{
    float coef[20]; //ARRAY DI COEFFICIENTI
    float x;
    int gr,i;
    printf("PROGRAMMA CHE GENERA UN POLINOMIO E NE CALCOLA IL VALORE CON
        ALGORITMO DI HORNER\n\n");
    printf("Inserisci il grado del polinomio : ");
    scanf("%d",&gr); //MEMORIZZA GRADO
    for(i=gr;i>=0;i--)
    {
        printf("\nImmetti il coefficiente della x^%d: ",i);
        scanf("%f",&coef[i]); //MEMORIZZA COEFFICIENTI NELL'ARREY
    }
    printf("\nImmetti il valore della x : ");
    scanf("%f",&x);
    printf("\nIl valore del polinomio e': %f\n\n",horner(gr,coef,x));
}

float horner(int gr, float coef[], float x)
{
    int i;
    float y=0.0;
    for(i=gr;i>=0;i--)
        y=y*x+coef[i]; //CALCOLO DEL VALORE DEL POLINOMIO
    return y;
}
```



```
C:\Windows\system32\cmd.exe
PROGRAMMA CHE GENERA UN POLINOMIO E NE CALCOLA IL VALORE CON ALGORITMO DI HOR
Inserisci il grado del polinomio : 3
Immetti il coefficiente della x^3: 4
Immetti il coefficiente della x^2: 5
Immetti il coefficiente della x^1: 6
Immetti il coefficiente della x^0: 7
Immetti il valore della x : 2
Il valore del polinomio e': 71.000000
Premere un tasto per continuare . . . _
```

2.**[liv.1]** Scrivere una function C per calcolare una somma di molti addendi mediante l'algoritmo del raddoppiamento ricorsivo (batch adding - versione iterativa).

```
#include <stdio.h>
#include <math.h>
int somma_radd(int a[], int n); //SOMMA PER RADDOPPIAMENTO
void main()
{
    int n=8,i;
    int a[8];
    printf("PROGRAMMA CHE SOMMA MOLTI ADDENDI TRAMITE RADDOPPIAMENTO\n\n");
    printf("RICORSIVO\n\n");
    printf("Inserire gli addendi\n");
    for(i=0;i<n;i++)
    {
        printf("\n%d elemento: ",i+1);
        scanf("%d",&a[i]); //SI ACQUISISCONO GLI ELEMENTI DELL'ARRAY
    }
    printf("\nLa somma degli addendi e': %d\n\n",somma_radd(a,n-1));
}

int somma_radd(int a[], int n)
{
    if(n>0)
    {
        a[n-1]=a[n]+a[n-1]; //SI SOMMANO I DUE ELEMENTI VICINI NELL'ARRAY
        return somma_radd(a,n-1); //RESTITUISCE LA SOMMA PARZIALE
    }
    if(n<=0)
    {
        return a[n]; //RESTITUISCE LA SOMMA TOTALE
    }
}
```

```
C:\Windows\system32\cmd.exe
PROGRAMMA CHE SOMMA MOLTI ADDENDI TRAMITE RADDOPPIAMENTO RICORSIVO
Inserire gli addendi
1o elemento: 0
2o elemento: 1
3o elemento: 1
4o elemento: 2
5o elemento: 3
6o elemento: 5
7o elemento: 8
8o elemento: 13
La somma degli addendi e': 33
Premere un tasto per continuare . . .
```

3.*[liv.1]* Scrivere una function C per calcolare una somma con il criterio di arresto naturale.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <float.h>
double sommatoriaD_iterativa(int num); //FUNCTION DI ARRESTO NATURALE
int main()
{
    int n;
    printf("PROGRAMMA CHE ATTUA LA SOMMA CON ARRESTO NATURALE\n\n");
    printf("Inserisci il valore delle somme da fare: ");
    scanf("%d", &n);
    printf("\n\nSomma: %22.16e\n", sommatoriaD_iterativa(n));
    printf("\n\n");
}
double sommatoriaD_iterativa(int num)
{
    double S=0;
    int k=1;
    while (k<=num && pow(1.f/k,2)>=FLT_EPSILON*S) //CONDIZIONE DI ARRESTO
    {
        S=S+(double)pow(1.f/k,2); //SOMMA DEGLI ADDENDI
        k++; //CONTATORE PASSI
    }
    printf("\n\nIl numero di passi e': %d", k-1);
    return(S);
}
```

```
C:\Windows\system32\cmd.exe
PROGRAMMA CHE ATTUA LA SOMMA CON ARRESTO NATURALE
Inserisci il valore delle somme da fare: 4

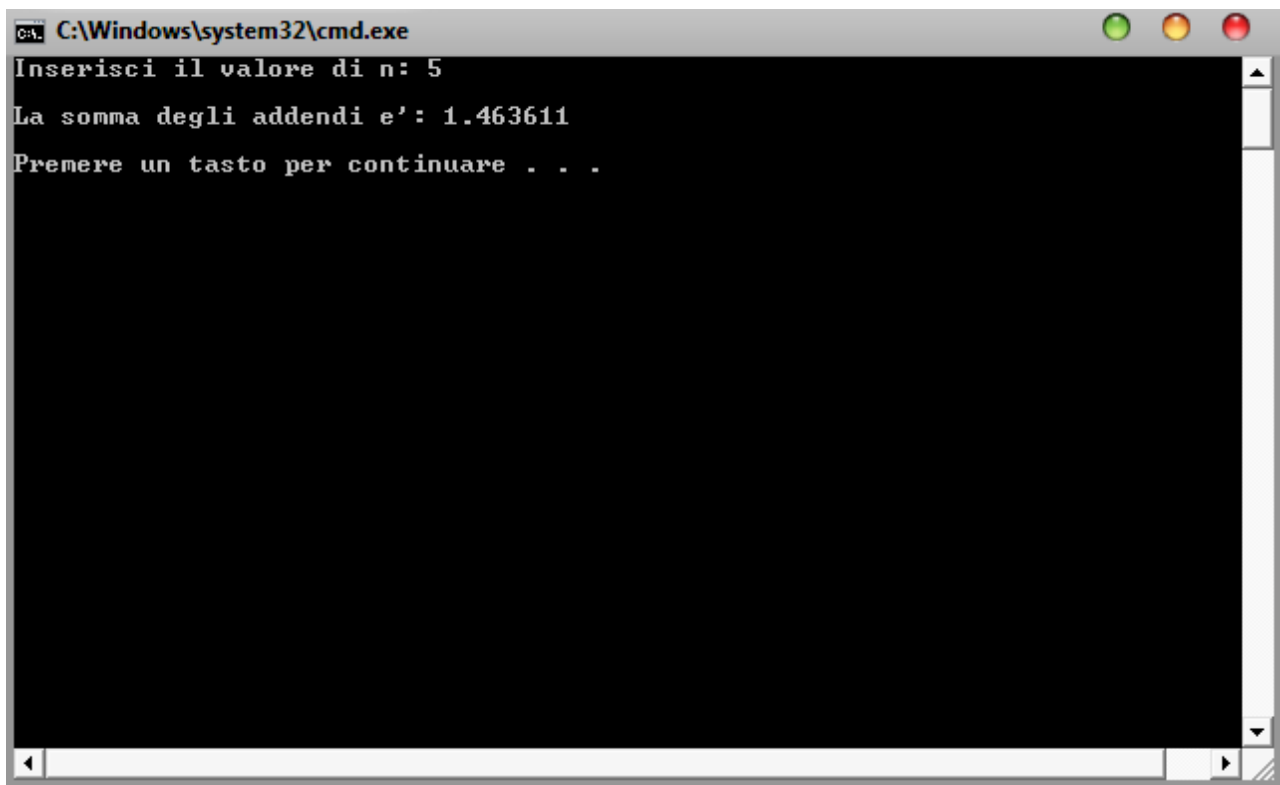
Il numero di passi e': 4
Somma: 1.4236111111111112e+000

Premere un tasto per continuare . . . _
```

4.[liv.1] Scrivere una function C per calcolare una somma di addendi ordinati

```
#include<stdio.h>
#include<math.h>
float somma(int n);
void main()
{
    int n;
    printf("Inserisci il valore di n: ");
    scanf("%d",&n);//SI DA IN INPUT IL NUMERO DEGLI ADDENDI
    printf("\nLa somma degli addendi e': %f\n",somma(n));
}

float somma(int n)
{
    float sum=0;
    int k;
    for(k=1;k<=n;k++)//SOMMA DECRESCENTE
        sum=sum+(1/(float)(pow(k,2)));
    return sum;
}
```



```
C:\Windows\system32\cmd.exe
Inserisci il valore di n: 5
La somma degli addendi e': 1.463611
Premere un tasto per continuare . . .
```

P2_04_01_AC

1. *[liv.1] Confrontando i risultati con quelli delle relative funzioni del C ed utilizzando per le stringhe*

- o *l'allocazione statica*
- o *l'allocazione dinamica*

scrivere una function C che legga i singoli caratteri costruendo la stringa che li contiene senza usare `strcat(...)`.

STATICA:

```
#include <stdio.h>
#include <string.h>
void concatena(char stringa[]);
#define n 64
int main(void){
    char stringa[n];
    printf("Inserire la stringa\t ");
    concatena(stringa);
```

```

        printf("La stringa e [%s], di lunghezza %d\n",stringa,strlen(stringa));
        getchar();
return 0;
}

void concatena(char stringa[]){
    char c;
    short j=0;
    do{
        stringa[j++]=c=getchar();
    } while(c!='\n');
}

```

```

C:\Windows\system32\cmd.exe
Inserire la stringa      Il mi nome e Alessio Piromallo e questo eser. e per
same di Programmazione 2

La stringa e [Il mi nome e Alessio Piromallo e questo eser. e per l'esame di
grammazione 2
], di lunghezza 81

```

DINAMICO:

```

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void concatena(char *stringa,short len);

int main(void){
    char *stringa;//Puntatore a stringa
    short len=0;
    printf("Inserisci quanto lunga deve essere la stringa...");
    scanf("%d",&len);
    stringa=(char *) malloc (len+1);//alloca la memoria a stringa
    printf("\nInserire la stringa singolarmente\t ");
    concatena(stringa,len);
    printf("La stringa e ");
    puts(stringa);
    printf("\n\n");
    free(stringa);//Libera memoria inutilizzata
return 0;

```

```

}

void concatena(char *stringa,short len){
    short j;
    char c;
    for(j=0;j<len;j++)
    {
        scanf("%c",&c);
        //Al passo j il valore di "c" va inserito in stringa[j]
        *(stringa+j)=c;
        fflush(stdin);
    }

    //viene inserito a stringa[len+1] il terminatore "\0"
    *(stringa+j)='\0';
}

```

```

C:\Windows\system32\cmd.exe
Inserisci quanto lunga deve essere la stringa....18
Inserire la stringa singolarmente      A
l
e
s
s
i
o
P
i
r
o
m
a
l
l
o
La stringa e
Alessio Piromallo

Premere un tasto per continuare . . . _

```

2. *[liv.1] Confrontando i risultati con quelli delle relative funzioni del C ed utilizzando per le stringhe*

- o *l'allocazione statica*
- o *l'allocazione dinamica*

scrivere una function C che restituisca la concatenazione di due stringhe date in input senza usare `strcat(...)`.

STATICA:

```

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
void concatena(char p1[],char p2[],char p_tot[],short n1,short n2,short n3);
int main(){
    char p1[25],p2[25],p_tot[50];
    short n1,n2,n3;
    printf("Inserire la prima stringa\n");
    scanf("%s",&p1);

```

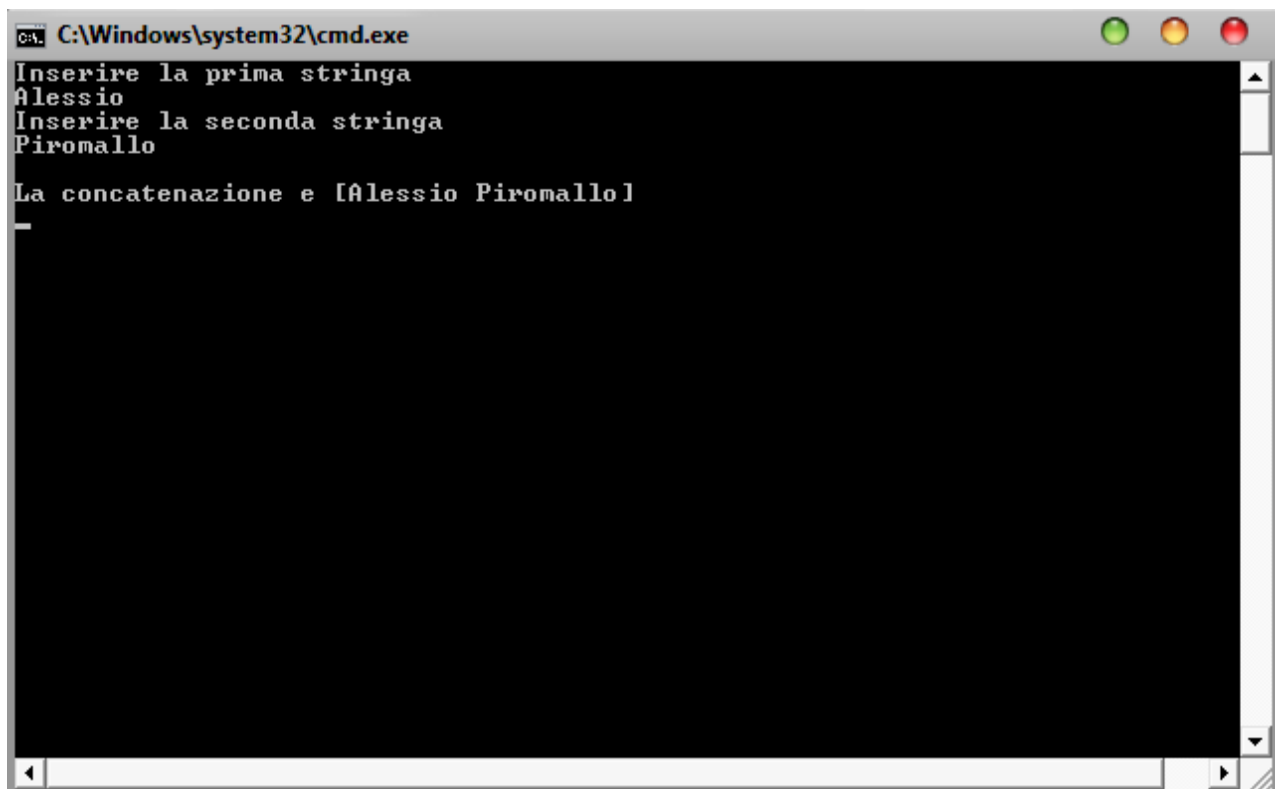


```

fflush(stdin);
printf("Inserire la seconda stringa\n");
scanf("%s",&p2);
fflush(stdin);
n1=strlen(p1);
n2=strlen(p2);
n3=n1+n2;
concatena(p1,p2,p_tot,n1,n2,n3);
printf("\nLa concatenazione e [%s]",p_tot);
printf("\n");
getchar();
return 0;}

void concatena(char p1[],char p2[],char p_tot[],short n1,short n2,short n3)
{
    short j=-1,y=-1,d=-1;
    do{
        if(j<n1-1){
            y++;j++;
            p_tot[y]=p1[j];}
        if(j==n1-1){
            j++;y++;
            p_tot[y]=' ';}
        if(j>=n1){
            y++;d++;
            p_tot[y]=p2[d];}
    } while(y<=n3);
}

```



A screenshot of a Windows command prompt window. The title bar at the top shows the file name "C:\Windows\system32\cmd.exe" and three standard window control buttons (minimize, maximize, close) on the right. The command prompt has a black background with white text. The text displayed is as follows: "Inserire la prima stringa", "Alessio", "Inserire la seconda stringa", "Piromallo", and "La concatenazione e [Alessio Piromallo]". A horizontal cursor line is positioned below the last line of text. The window has a standard Windows XP-style border and a scrollbar on the right side.

```
C:\Windows\system32\cmd.exe
Inserire la prima stringa
Alessio
Inserire la seconda stringa
Piromallo
La concatenazione e [Alessio Piromallo]
_
```

DINAMICA:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
void concatena(char *p1,char *p2,char *p_tot,short n1,short n2,short n3);

int main(void){
    char *p1,*p2,*p_tot;
    short n1,n2,n3;
    //Inserisco la lunghezza della prima stringa e la inserisco
    //e con la malloc la alloco
    printf("Inserire la lunghezza della prima stringa....");
    scanf("%d",&n1);
    fflush(stdin);
    p1=(char *)malloc(n1+1);
    printf("Inserire la prima stringa\n");
    gets(p1);
    fflush(stdin);
    //Inserisco la lunghezza della seconda stringa e la inserisco
    //e con la malloc la alloco
    printf("Inserire la lunghezza della seconda stringa....");
    scanf("%d",&n2);
    fflush(stdin);
    p2=(char *)malloc(n2+1);
    printf("Inserire la seconda stringa\n");
    gets(p2);
    fflush(stdin);
    //Con la malloc la alloco alloco la stringa che dovra
    //contenere i valori di p1 e p2
    n3=n1+n2;
    p_tot=(char *)malloc(n3+1);
    concatena(p1,p2,p_tot,n1,n2,n3);
    printf("\nLa concatenazione e [%s]",p_tot);
    printf("\n");
    getchar();
    //Libero la memoria allocata da p1,p2,p_tot
    free(p1);
    free(p2);
    free(p_tot);
return 0;}

void concatena(char *p1,char *p2,char *p_tot,short n1,short n2,short n3)
{
    short j=-1,y=-1,d=-1;
    do{
        // se inserisco la prima stringa nella stringa concatenativa
        if(j<n1-1){
            y++;j++;
            *(p_tot+y)=*(p1+j);}
        // inserisco in (j+1)=n1 uno spazzio tra le 2 frasi
        if(j==n1-1){
            j++;y++;
            *(p_tot+y)=' ';}
        // se inserisco la seconda stringa nella stringa concatenativa
        //partendo da j+2 rispetto la prima stringa
        if(j>=n1){
            y++;d++;
            *(p_tot+y)=*(p2+d);}
    }while(y<=n3);
}
```

C:\Windows\system32\cmd.exe

```
Inserire la lunghezza della prima stringa....9
Inserire la prima stringa
Piromallo
Inserire la lunghezza della seconda stringa....7
Inserire la seconda stringa
Alessio
```

```
La concatenazione e [Piromallo Alessio]
```

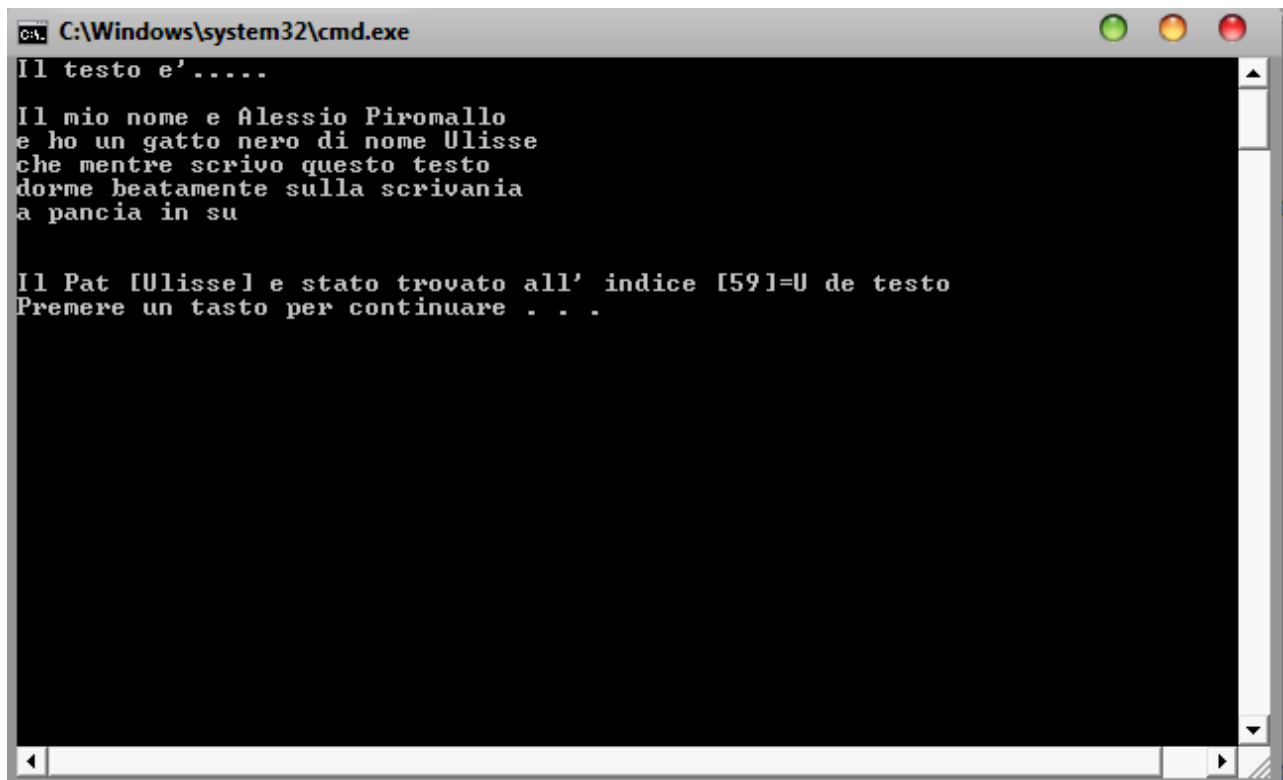
P2_04_02_AC

1.[liv.1] Confrontando i risultati con quelli delle relative funzioni del C, scrivere una function C che restituisca la prima occorrenza di una sottostringa in una stringa senza usare `strstr(...)`.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
short ricerca_pat(char *testo,char *pat,short n_txt,short n_pat);
int main(void){
char *testo="Il mio nome e Alessio Piromallo\n"
           "e ho un gatto nero di nome Ulisse\n"
           "che mentre scrivo questo testo\n"
           "dorme beatamente sulla scrivania\n"
           "a pancia in su\n";

char *pat="Ulisse";
short indice=0;
short n_txt=strlen(testo);
short n_pat=strlen(pat);
printf("Il testo e'..... \n\n%s",testo);
indice=ricerca_pat(testo,pat,n_txt,n_pat);
printf("\n\nIl Pat [%s] e stato trovato all' indice [%d]=%c de testo\n",pat,indice,*(&testo[indice]));
return 0;
}

short ricerca_pat(char *testo,char *pat,short n_txt,short n_pat){
    short indice=0,flag=0,x=0,j=0,y=j;
    do{
        //in caso che testo[j]coincide conpat[x]
        //i due indici vengono incrementati
        //e il flag aumenta di 1
        if(*(&testo[j])==*(pat+x))
        {
            flag++;
            j++;x++;
        }
        //se i caratteri precedenti erano simili ma non si completa il pat
        //xche diverso da quello ricercato l'indice del pat torna a 0
        //l'indice del testo aumenta e il flag torna a 0
        if(*(&testo[j])!=*(pat+x) && flag<n_pat)
        {
            flag=0;
            j++;x=0;
        }
        //se il pat viene trovato flag = strlen(pat)
        //allora rereturn la posizione iniziale
        //del indice indice testo- lunghezza pat
        if(flag==n_pat)
        {
            return indice=j-n_pat;
        }
    }while(j<n_txt);
    return 0;
}
```



```
C:\Windows\system32\cmd.exe
Il testo e'.....
Il mio nome e Alessio Piromallo
e ho un gatto nero di nome Ulisse
che mentre scrivo questo testo
dorme beatamente sulla scrivania
a pancia in su

Il Pat [Ulisse] e stato trovato all' indice [59]=U de testo
Premere un tasto per continuare . . .
```

2.[liv.2] Usando l'allocazione dinamica e le funzioni del C per manipolare le stringhe, scrivere una function C che restituisca la posizione di tutte le occorrenze di una sottostringa in una stringa ed il loro numero totale.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
short ricerca_pat(short ind[10],char *testo,char *pat,short n_txt,short n_pat);
int main(void){
char *testo="Il mio nome e Alessio Piromallo\n"
           "e ho un gatto nero di nome Ulisse\n"
           "che mentre scrivo questo testo\n"
           "dorme beatamente sulla scrivania\n"
           "a pancia in su\n";

char *pat="nome";
short indice=0;
short n_txt=strlen(testo);
short n_pat=strlen(pat);
short ind[10],k;
printf("Il testo e'.... \n\n%s",testo);
indice=ricerca_pat(ind,testo,pat,n_txt,n_pat);
for(k=0;k<indice;k++)
//L'indice delle occorrenze e salvato nell array ind[] quindi per
//estrarlo bisogna puntare al test con indice l' indice al valore di ind
printf("\n\nIl Pat [%s] e stato trovato all' indice [%d]=%c de testo\n",pat,ind[k],*(testo+ind[k]));
return 0;
}

short ricerca_pat(short ind[10],char *testo,char *pat,short n_txt,short n_pat){
    short d=0,indice=0,flag=0,x=0,j=0,y=j;
    do{
        //in caso che testo[j]coincide conpat[x]
        //i due indici vengono incrementati
        //e il flag aumenta di 1
        if(*(testo+j)==*(pat+x))
        {
            flag++;j++;x++;
        }
        //se i caratteri precedenti erano simili ma non si completa il pat
        //xche diverso da quello ricercato l'indice del pat torna a 0
        //l'indice del testo aumenta e il flag torna a 0
        if(*(testo+j)!=*(pat+x) && flag<n_pat)
        {
            flag=0; j++;x=0;
        }
        //se il pat viene trovato flag = strlen(pat)
        //allora viene salvato nell array ind l'indice
        //viene incrementato l'indice dell array,
        //viene riazzerato l'indice pat e incrementato j
        if(flag==n_pat)
        {
            ind[d]=j-n_pat;
            d++;x=0;j++;flag=0;
        }
    } while(j<n_txt);
    return d;
}
```

cmd C:\Windows\system32\cmd.exe

Il testo e'.....

Il mio nome e Alessio Piromallo
e ho un gatto nero di nome Ulisse
che mentre scrivo questo testo
dorme beatamente sulla scrivania
a pancia in su

Il Pat [nome] e stato trovato all' indice [7]=n de testo

Il Pat [nome] e stato trovato all' indice [54]=n de testo
Premere un tasto per continuare . . . _

3. [liv.1] Utilizzando per le stringhe

o l'allocazione statica

o l'allocazione dinamica

scrivere una function C che elimini tutte le occorrenze di una data sottostringa in una stringa.

STATICA:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
short trova(char testo[],char pat_1[],short indice[30],short n_txt,short n_pat_1);
void elimina(short i,char testo[],char pat_1[],short indice[],short n_txt,short n_pat_1);
int main(void){
    char testo[50]="casa mia e grande come la mia casa al mare";
    char pat_1[10]="mia";
    short n_txt,n_pat_1,i=0,j,indice[30];
    n_txt=strlen(testo),n_pat_1=strlen(pat_1);
    i=trova(testo,pat_1,indice,n_txt,n_pat_1);
    printf("\nIl testo originale e .... %s\n\nIl pat originale e [%s]\n",testo,pat_1);
    for(j=0;j<i;j++){
        {printf("\nIl Pat e stato trovato all indice [%d]=%c\n",indice[j],testo[indice[j]]);}
        elimina(i,testo,pat_1,indice,n_txt,n_pat_1);
        printf("\nIl testo modificato e .... %s\n\nIl pat originale e [%s]\n",testo,pat_1);
    }
    return 0;}

short trova(char testo[40],char pat_1[4],short indice[30],short n_txt,short n_pat_1){
    short flag=0,j=0,x=0,i=0;
    do{
        //viene trovato l'indice della stringa cercata e inserita nell array indice
        if(flag==n_pat_1)
        { indice[i]=(j-n_pat_1);
          flag=0;j++;x=0,i++;}

        if(testo[j]==pat_1[x])
        { flag++;j++;x++;}

        if(testo[j]!=pat_1[x]&& flag < n_pat_1)
        { flag=0;j++;x=0;}

    } while(j<=n_txt);
    return i; }

void elimina(short i,char testo[],char pat_1[],short indice[],short n_txt,short n_pat_1)
{
    short i_pat,len_pat,x=0,j;
    //scorre la stringa tante volte quante sono presenti i pat nella stringa
    for(i_pat=0;i_pat<i;i_pat++)
    {
        //scorre la stringa tante volte quant'è lunga il pat
        for(len_pat=0;len_pat<=n_pat_1;len_pat++)
        {
            j=indice[x];
            //sposta i valori successivo al passo attuale ne passo attuale
            //stringe la stringa
            do{ testo[j]=testo[j+1];
              j++;
            } while(j<n_txt); }
        //viene ricalcolato l'indice del pat succesivo
        //in base alle modifiche effettuate sulla stringa dopo il ridimensionamento
        x++;
        indice[x]=(indice[x]-n_pat_1)-1;
    }
}
```

C:\Windows\system32\cmd.exe

Il testo originale e casa mia e grande come la mia casa al mare

Il pat originale e [mia]

Il Pat e stato trovato all indice [5]=m

Il Pat e stato trovato all indice [26]=m

Il testo modificato e casa e grande come la casa al mare

Il pat originale e [mia]

Premere un tasto per continuare . . . _

DINAMICA:

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int uguale(char*,char*,int); //TROVA LE OCCORRENZE
void main()
{
    char *testo;
    char *pattern;
    int len_p,len_t,i,j;
    printf("Inserisci la lunghezza del testo e il testo stesso\n\n");
    scanf("%d",&len_t);
    len_t++;
    testo=(char *)malloc(len_t); //SI ALLOCA DINAMICAMENTE SPAZIO PER IL TESTO
    fflush(stdin);
    gets(testo);
    printf("Il testo e' \t%s\n\n",testo);
    len_t=strlen(testo);
    //-----
    printf("Inserisci la lunghezza del pattern e il pattern stesso\n\n");
    scanf("%d",&len_p);
    len_p++;
    pattern=(char *)malloc(len_p); //SI ALLOCA SPAZIO PER IL PATTERN
    fflush(stdin);
    gets(pattern);
    printf("La sottostringa e' \t%s\n\n",pattern);
    len_p=strlen(pattern);

    for(i=0;i<len_t;i++)
    { if(uguale(pattern,(testo+i),len_p))
        {
            //si trovano le occorrenze e si stringe la stringa per eliminarle
            for(j=i;j<len_t;j++)
            {
                *(testo+j)=*(testo+(j+len_p));
            }
        }
    }
    printf("\n\nLa stringa senza occorrenze:\t%s\n\n",testo);
}

int uguale(char *str,char *pat,int n)
{
    int j=0;
    do
    {
        j++;
    }
    while(*(str+(j-1))==*(pat+(j-1)) && j<n);
    //vede se il pattern viene trovato in ogni suo carattere
    if(*(str+(j-1))==*(pat+(j-1)) && j==n)
        return 1;
    else
        return 0;
}
```

```
C:\Windows\system32\cmd.exe
Inserisci la lunghezza del testo e il testo stesso
30
il mio gatto si chiama ulisse
Il testo e'      il mio gatto si chiama ulisse

Inserisci la lunghezza del pattern  e il pattern stesso
6
ulisse
La sottostringa e'      ulisse

La stringa senza occorrenze:      il mio gatto si chiama
Premere un tasto per continuare . . .
```

4. [liv.1] Utilizzando per le stringhe

o l'allocazione statica

o l'allocazione dinamica

scrivere una function C che sostituisca in un testo tutte le occorrenze di una data sottostringa S1 con un'altra S2 (le due sottostringhe possono avere anche lunghezze diverse).

STATICA:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
short trova(char testo[],char pat_1[],short indice[30],short n_txt,short n_pat_1);
void scambia(short i,char testo[],char pat_1[],char pat_2[],short indice[],short n_txt,short n_pat_1,short n_pat_2);
int main(void){
    char testo[50];
    char pat_1[10];
    char pat_2[10];
    short n_txt,n_pat_1,n_pat_2,i=0,j;
    short indice[30];
    printf("\nInserire il Testo da Analizzare....\n");
    gets(testo); fflush(stdin);
    printf("\nInserire il Pat da Trovare....\t");
    gets(pat_1); fflush(stdin);
    printf("\nInserire il Pat da Scambiare....\t");
    gets(pat_2); fflush(stdin);
    n_txt=strlen(testo),n_pat_1=strlen(pat_1),n_pat_2=strlen(pat_2),i=0,j;
    i=trova(testo,pat_1,indice,n_txt,n_pat_1);
    for(j=0;j<i;j++){
        printf("\nIl Pat e stato trovato all indice [%d]=%c\n",indice[j],testo[indice[j]]);}
    scambia(i,testo, pat_1, pat_2, indice, n_txt, n_pat_1, n_pat_2);
    printf("\nIl testo modificato e .... %s \n\nIl pat originale e [%s] e il sostitutivo e [%s]\n",testo,pat_1,pat_2);
    //free(testo); free(pat_1); free(pat_2);
    return 0;
}

short trova(char testo[40],char pat_1[4],short indice[30],short n_txt,short n_pat_1){
    short flag=0,j=0,x=0,i=0;
    do{
        //viene trovato l'indice della stringa cercata e inserita nell array indice
        if(flag==n_pat_1)
        { indice[i]=(j-n_pat_1);
          flag=0;j++;x=0,i++;}

        if(testo[j]==pat_1[x])
        { flag++;j++;x++;}

        if(testo[j]!=pat_1[x]&& flag < n_pat_1)
        { flag=0;j++;x=0;}

    } while(j<=n_txt);
    return i;
}
```

```

void scambia(short i,char testo[30],char pat_1[3],char pat_2[4],short indice[30],short n_txt,short n_pat_1,short
n_pat_2){
    short x=0,j=0,flag_passi=0,flag_indici=0,u=0,i_m=0;
    short ind=indice[x];

    //se il pattern 2 ha la stessa lunghezza del pattern 1
    if(n_pat_2==n_pat_1)
    {
        do{
            do{
                //viene inserito il pat[j] nell testo con indice il
                //valore contenuto in indice[] trovato precedentemente
                testo[ind] = pat_2[j];
                ind++;j++;flag_passi++;
            }while(flag_passi<n_pat_2);
            // quando il flag e uguale alla lunghezza del pat
            //allora la stringa ( nuova )e stata completamente
            //inserita a posto della precedente
            if(flag_passi==n_pat_2)
            { flag_indici++;flag_passi=0;x++;ind=indice[x];j=0;}
        }while(flag_indici<i);
    }

    //se il pattern 2 e piu lungo del pattern 1
    if(n_pat_2 > n_pat_1)
    {
        do{ //viene incrementato la grandezza dell array
            //per permettere l'inserimento dell pat 2
            //maggiore di quello da sostituire
            for(j=n_txt;j>=ind-1;j--)
            { testo[j+1]=testo[j];}
            j=0;
            do{
                testo[ind] = pat_2[j];
                ind++;j++;flag_passi++;
            }while(flag_passi<n_pat_2);
            if(flag_passi==n_pat_2)
            { flag_indici++;flag_passi=0;x++;ind=indice[x];j=0;}
        }while(flag_indici<i);
    }

    //se il pattern 2 e piu corto del pattern 1
    if(n_pat_2 < n_pat_1)
    {
        do{ //viene prima deciso quante volte accorciare
            //la stringa in base a quante volte il pat 1 e stata trovata
            for(i_m=0;i_m<(n_pat_1-n_pat_2);i_m++){
                //Viene decrementata la stringa per permettere l'inserimento dell pat 2
                for(j=ind;j<=n_txt;j++)
                { testo[j]=testo[j+1];}
                j=0;u++;}
            i_m=0;

            do{
                testo[ind] = pat_2[j];
                ind++;j++;flag_passi++;
            }while(flag_passi<n_pat_2);
            if(flag_passi==n_pat_2)
            { flag_indici++;flag_passi=0;x++;ind=indice[x];j=0;}
        }while(flag_indici<i);
    }
}

```

```
C:\Windows\system32\cmd.exe

Inserire il Testo da Analizzare....
Nel mezzo del cammin di nostra vita mi ritrovai per una selva oscura ch  la d
tta via era smarrita

Inserire il Pat da Trovare....  nostra
Inserire il Pat da Scambiare....      sua

Il Pat   stato trovato all'indice [24]=n
Il testo modificato   .... Nel mezzo del cammin di sua vita mi ritrovai per
selva oscura ch  la diritta via era smarrita

Il pat originale   [nostra] e il sostitutivo   [sua]
Premere un tasto per continuare . . .
```

DINAMICA:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
short trova(char *testo, char *pat_1, short indice[30], short n_txt, short n_pat_1);
void scambia(short i, char *testo, char *pat_1, char *pat_2, short indice[], short n_txt, short n_pat_1, short n_pat_2);
int main(void){
    char *testo, *pat_1, *pat_2;
    short indice[30], n_txt, n_pat_1, n_pat_2, i=0, j;
    printf("Lughezza testo\n"); scanf("%d", &n_txt); fflush(stdin);
    printf("Lughezza Pat1\n"); scanf("%d", &n_pat_1); fflush(stdin);
    printf("Lughezza Pat2\n"); scanf("%d", &n_pat_2); fflush(stdin);
    testo=(char *)malloc(n_txt); pat_1=(char *)malloc(n_pat_1); pat_2=(char *)malloc(n_pat_2);
    printf("\nInserire il Testo da Analizzare....\n"); gets(testo); fflush(stdin);
    printf("\nInserire il Pat da Trovare....\n"); gets(pat_1); fflush(stdin);
    printf("\nInserire il Pat da Scambiare....\n"); gets(pat_2); fflush(stdin);
    i=trova(testo, pat_1, indice, n_txt, n_pat_1);
    for(j=0; j<i; j++){
        {printf("\nIl Pat e stato trovato all indice [%d]=%c\n", indice[j], *(testo+(indice[j])));}
        scambia(j, testo, pat_1, pat_2, indice, n_txt, n_pat_1, n_pat_2);
        printf("\nIl testo modificato e .... %s \nIl pat originale e [%s] e il sostitutivo e [%s]\n", testo, pat_1, pat_2);
        free(testo); free(pat_1); free(pat_2);
    }
    return 0;}

short trova(char *testo, char *pat_1, short indice[30], short n_txt, short n_pat_1){
    short flag=0, j=0, x=0, i=0;
    do{ //viene trovato l'indice della stringa cercata e inserita nell array indice
        if(flag==n_pat_1)
            { indice[i]=(j-n_pat_1);
              flag=0; j++; x=0, i++; }

        if(*(testo+j)==*(pat_1+x))
            { flag++; j++; x++; }

        if(*(testo+j)!=*(pat_1+x) && flag < n_pat_1)
            { flag=0; j++; x=0; }

    } while(j<=n_txt);
    return i;
}

void scambia(short i, char *testo, char *pat_1, char *pat_2, short indice[30], short n_txt, short n_pat_1, short n_pat_2){
    short x=0, j=0, flag_passi=0, flag_indici=0, u=0, i_m=0;
    short ind=indice[x];
    //se il pattern 2 ha la stessa lunghezza del pattern 1
    if(n_pat_2==n_pat_1)
    {
        do{
            do{ //viene inserito il pat[j] nell testo con indice il
                //valore contenuto in indice[] trovato precedentemente
                *(testo+ind) = *(pat_2+j);
                ind++; j++; flag_passi++;
            } while(flag_passi<n_pat_2);
            // quando il flag e uguale alla lunghezza del pat
            //allora la stringa ( nuova )e stata completamente
            //inserita a posto della precedente
            if(flag_passi==n_pat_2)
            { flag_indici++; flag_passi=0; x++; ind=indice[x]; j=0; }
        } while(flag_indici<i);
    }
}
```



```

//se il pattern 2 e piu lungo del pattern 1
if(n_pat_2 > n_pat_1)
{
    do{ //viene incrementato la grandezza dell array
        //per permettere l'inserimento dell pat 2
        //maggiore di quello da sostituire
        for(j=n_txt;j>=ind-1;j--){
            *(testo+(j+1))= *(testo+j);}
        j=0;
        do{
            *(testo+ind) = *(pat_2+j);
            ind++;j++;flag_passi++;
        }while(flag_passi<n_pat_2);
        if(flag_passi==n_pat_2)
        { flag_indici++;flag_passi=0;x++;ind=indice[x];j=0;}
    }while(flag_indici<i);
}

//se il pattern 2 e piu corto del pattern 1
if(n_pat_2 < n_pat_1)
{
    do{ //viene prima deciso quante volte accorciare
        //la stringa in base a quante volte il pat 1 e stata trovata
        for(i_m=0;i_m<(n_pat_1-n_pat_2);i_m++){
            //Viene decrementata la stringa per permettere l'inserimento dell pat 2
            for(j=ind;j<=n_txt;j++){
                *(testo+j)=*(testo+(j+1));}
            j=0;u++;}
        i_m=0;

        do{
            *(testo+ind) = *(pat_2+j);
            ind++;j++;flag_passi++;
        }while(flag_passi<n_pat_2);
        if(flag_passi==n_pat_2)
        { flag_indici++;flag_passi=0;x++;ind=indice[x];j=0;}
    }while(flag_indici<i);
}
}

```

```

C:\Windows\system32\cmd.exe
Lughezza testo
48
Lughezza Pat1
7
Lughezza Pat2
7

Inserire il Testo da Analizzare....
il mio nome e Alessio Piromallo e vivo a Portici

Inserire il Pat da Trovare....  Portici

Inserire il Pat da Scambiare....      Ginevra

Il Pat e stato trovato all indice [41]=P

Il testo modificato e ....  il mio nome e Alessio Piromallo e vivo a Ginevra

Il pat originale e [Portici] e il sostitutivo e [Ginevra]
Premere un tasto per continuare

```

P2_05_02_C

1.**[liv.1]** A partire da una matrice $A(m \times n)$, del tipo sotto indicato, allocata staticamente [risp. dinamicamente per righe] visualizzarne gli elementi per colonne

$A(4 \times 6)$

/	11	12	13	14	15	16	/
/	21	22	23	24	25	26	/
/	31	32	33	34	35	36	/
/	41	42	43	44	45	46	/

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define RIGA 4
#define COLONNA 6

int main(void)
{
    int *a; short i,j;
    srand((unsigned) time (NULL));
    a= (int *)malloc(RIGA*COLONNA*sizeof (int));
    printf("\nMatrice statica A visualizzata per righe:\n\n");
    for (i=0;i<RIGA;i++)
    {
        for(j=0;j<COLONNA;j++)
        {
            *(a+i*RIGA+j)=11+rand()*36/RAND_MAX;
            printf("%d\t",*(a+i*RIGA+j));
        }
        printf("\n\n");
    }
    printf("La matrice A visualizzata per colonne diventa:\n\n");
    for(j=0;j<COLONNA;j++)
    {
        for(i=0;i<RIGA;i++)
        {
            printf("%d ",*(a+i*RIGA+j));
        }
    }
    printf("\n");
    return 0;
}
```

```
C:\Windows\system32\cmd.exe

Matrice statica A visualizzata per righe:
41      12      24      19      27      36
18      35      26      11      21      34
35      46      26      15      12      38
14      45      20      24      24      26

La matrice A visualizzata per colonne diventa:
41 18 35 14 12 35 46 45 24 26 26 20 19 11 15 24 18 35 14 24 35 46 45 26
Premere un tasto per continuare . . . _
```

P2_05_03_C

1.[liv.1] Scrivere una function C che restituisca la matrice C prodotto righe×colonne [vedi pdf delle dispense] di due matrici rettangolari A e B le cui dimensioni sono stabilite in input (usare per tutte le matrici l'allocazione dinamica e generarle come numeri reali random). C'è qualche preferenza nell'usare **malloc()** o **calloc()** rispettivamente per A, B o C? Verificare se i tempi di esecuzione sono gli stessi.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void prodotto (int,int,int,int *,int *,int *); //FUNZIONE CHE CALCOLA LA MATRICE PRODOTTO
void tempo(void); //CALCOLA TEMPI DI ESECUZIONE

int main (void)
{
    int i,j,m,p,n;
    int *a,*b,*c; //LE TRE MATRICI DICHIARATE DINAMICAMENTE
    srand ((unsigned)time (NULL));
    a=(int *)calloc (100,sizeof (int));
    b=(int *)calloc (100,sizeof (int));
    c=(int *)malloc (sizeof (int));
    printf("Inserire il numero di righe della matrice A: ");
    scanf("%d",&m);
    printf("Inserire il numero di colonne della matrice A che corrisponderanno anche alle righe di B: ");
    //il prodotto righe per colonne presuppone che il numero di
    //colonne della prima matrice sia uguale al numero di righe della seconda
    scanf("%d",&p);
    printf("Inserire il numero di colonne della matrice B: ");
    scanf("%d",&n);
    printf("\nMatrice A\n\n"); //SI INIZIALIZZA LA MATRICE A
    for (i=0;i<m;i++)
    {
        for (j=0;j<p;j++)
        {
            *(a+i*p+j)=rand()*10/RAND_MAX;
            printf("%d\t",*(a+i*p+j));
        }
        printf("\n");
    }
    printf("\nMatrice B\n\n"); //MATRICE B
    for (i=0;i<p;i++)
    {
        for (j=0;j<n;j++)
        {
            *(b+i*n+j)=rand()*10/RAND_MAX;
            printf("%d\t",*(b+i*n+j));
        }
        printf("\n");
    }
    prodotto (m,p,n,a,b,c);
    tempo();
    return 0;
}
```

```

void prodotto (int m,int p,int n,int *a,int *b,int *c)
{
    int x,y,i,j,k;
    x=0; y=0;
    for (k=0;k<m;k++)//SCORRE LE RIGHE DI A
    {
        for (i=0;i<n;i++)//SCORRE LE COLONNE DI B
        {
            for (j=0;j<p;j++)//SCORRE LE COLONNE DI A E LE RIGHE DI B
            {
                //PRODOTTO TRA L'ELEMENTO A[K][J] E L'ELEMENTO B[J][I]
                y=*(a+k*p+j)*(*(b+j*p+i));
                x=x+y;//SOMMA DEI PRODOTTI CON VARIABILI D'APPOGGIO
            }
            //ASSEGNAZIONE ALLA MATRICE C DELLA SOMMA DEI PRODOTTI
            *(c+k*p+i)=x;
            x=0;
            y=0;
        }
    }
    printf("\nMatrice C\n\n");//VISUALIZZAZIONE DELLA MATRICE C

    for (i=0;i<m;i++)
    {
        for (j=0;j<p;j++)
        {
            printf("%d\t",*(c+i*p+j));
        }
        printf("\n");
    }
}

void tempo(void)
{
    time_t Time_start, Time_finish;
    unsigned long loop, Loop_max;
    double z, w, result, Telapsed_time, Telapsed_mean;
    time( &Time_start );

    printf( "Time in seconds since UTC 1/1/70:\t%ld\n", Time_start );
    printf( "UNIX time and date:\t\t\t%s", ctime( &Time_start ) );

    //Loop_max=(long)(pow(2,31)-1);
    Loop_max=100000000;
    printf( "Multiplying 2 floating point numbers %ld times...\n",Loop_max);
    for( loop = 0; loop < Loop_max; loop++ )
    {
        z=(double)rand(); w=(double)rand();
        result = z * w;
    }

    time( &Time_finish );

    printf( "Time in seconds since UTC 1/1/70:\t%ld\n", Time_finish );
    printf( "UNIX time and date:\t\t\t%s", ctime( &Time_finish ) );

    Telapsed_time = difftime( Time_finish, Time_start );

    Telapsed_mean = Telapsed_time/(double)Loop_max;
    printf( "\nProgram takes %12.2f seconds.\n", Telapsed_time );
    printf( "\n      mean %12.8e seconds.\n", Telapsed_mean );
}

```

```
C:\Windows\system32\cmd.exe
Inserire il numero di righe della matrice A: 2
Inserire il numero di colonne della matrice A che corrisponderanno anche alle
righe di B: 3
Inserire il numero di colonne della matrice B: 3

Matrice A
9      4      5
9      0      8

Matrice B
3      4      5
1      5      0
0      1      4

Matrice C
31      61      65
27      44      77
Time in seconds since UTC 1/1/70:      1243939213
Premere un tasto per continuare . . . _
```

P2_07_01_AC

1.[liv.1] Scrivere una function C che legga, mediante buffer di 200char, un file testo e lo visualizzi sullo schermo 40 char/riga e 25 righe/schermata.

```
#include <stdio.h>
#include <stdlib.h>
#define BUFSIZE 200
#define RIGHE_C 40
#define RIGHE_R 25
void testo(char nome_file[]);

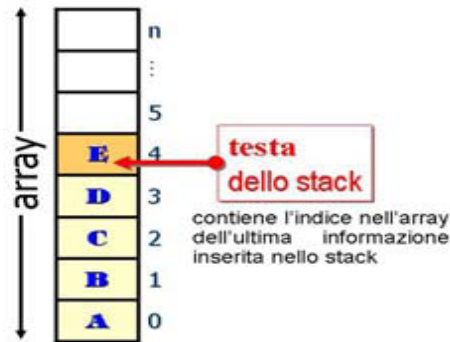
int main (void)
{
    char nome_file[10]="testo.txt";
    testo(nome_file);
    return 0;
}

void testo(char nome_file[10]){
    short r=0,c=0;
    char buffer[BUFSIZE],ch;
    FILE *fp;
    //se il file non viene trovato mostra un errore
    if((fp=fopen(nome_file,"r"))==NULL){
        printf("IL FILE NON E STATO APERTO\n");
        exit(1);}
    //se arriva a termine testo esce dal ciclo
    while(!feof(fp)){
        //controllo X le 25 righe
        while(r<RIGHE_R && !feof(fp)){
            //a ogni fine ciclo sottostante azzera il numero righe
            //x permettere di inserire la riga successiva
            c=0;
            //controllo X le 40 colonne
            while(c<RIGHE_C && !feof(fp)){
                //inseriesce il valore puntato attualmente in ch e poi nell array buffer
                ch=(char)getc(fp); buffer[c]=ch;
                //mostra il contenuto di buffer e incrementa "C" fino a 40
                putchar(buffer[c]); c++;
            }
            //incrementa la riga a fine colonna
            r++;
            printf("\t%d\n",feof(fp));
        }
        //nel caso che il testo non sia finito
        if(!feof(fp))
        {
            printf("\nLe 25 righe massime nella schermata sono state utilizzate\n");
            printf("Premere invio per visualizzare le rimanenti\n\n");
            getchar();
            r=0;
        }
    }
    //chiude il buffer del file testo.txt
    fclose(fp);
}
```


E E E E E E E E E E E E E E E

P2_08_03_T

1.[liv.1] Simulare in C la gestione di una pila (*stack*) tramite array statico (può essere anche un array di struct) creando le funzioni di manipolazione **push()** [inserimento] e **pop()** [eliminazione]. Il programma deve prevedere un menù che consenta di scegliere l'operazione da eseguire.



```
#include <stdlib.h>
#include <stdio.h>
#define n 9
void pop(short stach[n],short *i);
void push(short stach[],short *i);

int main(void){
    short stach[n]={0,0,0,0,0,0,0,0,0};
    short scelta=9,j,i=-1;
    while(scelta!=0){
        printf("FAI LA TUA SCELTA\n");
        printf("1 Inserimeto\n2 Eliminazione\n3 Visualizzazione\n0 Esci\n");
        printf("SCELTA : ");
        //seleziona le varie opzioni nel menu
        scanf("%d",&scelta);
        fflush(stdin);
        if(scelta == 1){
            if(i==n){//ARRAY PIENO
                printf("PILA PIENA\n");}
            if((i+1)<=n){//INSERIMENTO
                push(stach,&i);} }

        if(scelta == 2){
            if(i<=0){//ARRAY VUOTO
                printf("PILA VUOTA\n");}
            if((i+1)>=0){
                pop(stach,&i);} }

        if(scelta==3){
            if(i>=0)
                for(j=0;j<=i;j++)
                    printf("[%d] ",stach[j]);

            if(i<0)
                printf("NULLA DA STAMPARE PILA VUOTA");}

        printf("\n\n");
    }

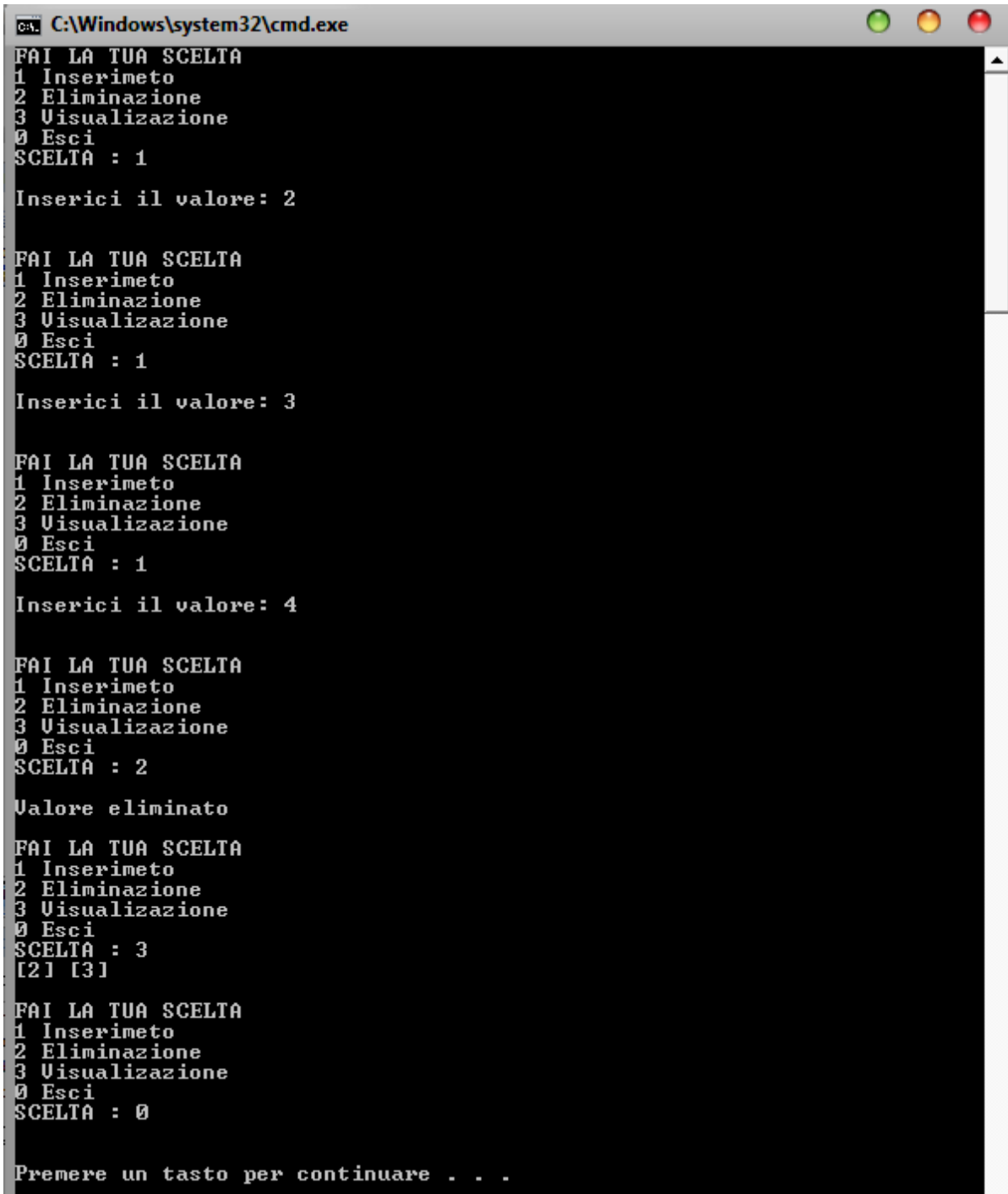
    return 0; }
```


//AGGIORNAMENTO DEL VALORE INDICE E INSERIMENTO NUOVO VALORE

```
void push(short stach[n],short *i){  
    printf("\nInserisci il valore: ");  
    *i=*i+1;  
    scanf("%d",&stach[*i]); fflush(stdin);}
```

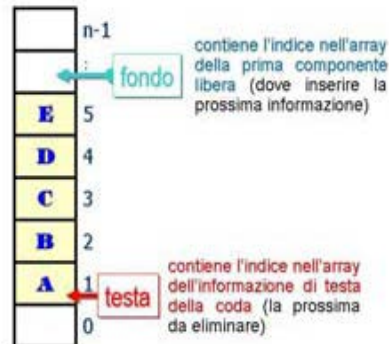
//AGGIORNAMENTO NUOVO INDICE I--

```
void pop(short stach[n],short *i){  
    printf("\nValore eliminato");  
    stach[*i]=0;  
    *i=*i-1;}
```



```
C:\Windows\system32\cmd.exe  
FAI LA TUA SCELTA  
1 Inserimento  
2 Eliminazione  
3 Visualizzazione  
0 Esci  
SCELTA : 1  
  
Inserisci il valore: 2  
  
FAI LA TUA SCELTA  
1 Inserimento  
2 Eliminazione  
3 Visualizzazione  
0 Esci  
SCELTA : 1  
  
Inserisci il valore: 3  
  
FAI LA TUA SCELTA  
1 Inserimento  
2 Eliminazione  
3 Visualizzazione  
0 Esci  
SCELTA : 1  
  
Inserisci il valore: 4  
  
FAI LA TUA SCELTA  
1 Inserimento  
2 Eliminazione  
3 Visualizzazione  
0 Esci  
SCELTA : 2  
  
Valore eliminato  
  
FAI LA TUA SCELTA  
1 Inserimento  
2 Eliminazione  
3 Visualizzazione  
0 Esci  
SCELTA : 3  
[2] [3]  
  
FAI LA TUA SCELTA  
1 Inserimento  
2 Eliminazione  
3 Visualizzazione  
0 Esci  
SCELTA : 0  
  
Premere un tasto per continuare . . .
```

2.[liv.1] Simulare in C la gestione di una coda (queue) tramite array statico (può essere anche un array di struct) creando le funzioni di manipolazione **enqueue()** [inserimento] e **dequeue()** [eliminazione]. Il programma deve prevedere un menù che consenta di scegliere l'operazione da eseguire. **Le informazioni NON vanno spostate!**



```
#include <stdlib.h>
#include <stdio.h>
#define n 5
void pop(short stach[n],short *indice);
void push(short stach[],short *i);

int main(void){
    short stach[n]={0,0,0,0,0};
    short scelta=9,j,i=-1,indice=0;
    while(scelta!=0){
        printf("FAI LA TUA SCELTA\n");
        printf("1 Inserimento\n2 Eliminazione\n3 Visualizzazione\n0 Esci\n");
        printf("SCELTA : ");
        //seleziona le varie opzioni nel menu
        scanf("%d",&scelta);
        fflush(stdin);
        if(scelta == 1){
            if(i>=n){//ARRAY PIENO
                printf("CODA PIENA\n");}
            if((i+1)<=n){//INSERIMENTO
                push(stach,&i);}}

        if(scelta == 2){
            if(indice>n || i<=0){//ARRAY VUOTO
                printf("CODA VUOTA\n");}
            if((i+1)>=0){
                pop(stach,&indice);}}

        if(scelta==3){
            if(i>=0 && indice<n)
                for(j=indice;j<=i;j++)
                    printf("[%d] ",stach[j]);

            if(i<0 || indice>=n)
                printf("NULLA DA STAMPARE CODA VUOTA");}

        printf("\n\n");
    }
    return 0;
}
```

```
//AGGIORNAMENTO DEL VALORE INDICE E INSERIMENTO NUOVO VALORE
```

```
void push(short stach[n],short *i){  
    printf("\nInserisci il valore: ");  
    *i=*i+1;  
    scanf("%d",&stach[*i]); fflush(stdin);}
```

```
//AGGIORNAMENTO NUOVO INDICE I--
```

```
void pop(short stach[n],short *indice){  
    printf("\nValore eliminato");  
    stach[*indice]=0;  
    *indice=*indice+1;}
```

cmd C:\Windows\system32\cmd.exe

FAI LA TUA SCELTA

1 Inserimeto
2 Eliminazione
3 Visualizzazione
0 Esci

SCELTA : 1

Inserisci il valore: 2

FAI LA TUA SCELTA

1 Inserimeto
2 Eliminazione
3 Visualizzazione
0 Esci

SCELTA : 1

Inserisci il valore: 3

FAI LA TUA SCELTA

1 Inserimeto
2 Eliminazione
3 Visualizzazione
0 Esci

SCELTA : 1

Inserisci il valore: 4

FAI LA TUA SCELTA

1 Inserimeto
2 Eliminazione
3 Visualizzazione
0 Esci

SCELTA : 3

[2] [3] [4]

FAI LA TUA SCELTA

1 Inserimeto
2 Eliminazione
3 Visualizzazione
0 Esci

SCELTA : 2

Valore eliminato

FAI LA TUA SCELTA

1 Inserimeto
2 Eliminazione
3 Visualizzazione
0 Esci

SCELTA : 3

[3] [4]

FAI LA TUA SCELTA

1 Inserimeto
2 Eliminazione
3 Visualizzazione
0 Esci

SCELTA : 1

Inserisci il valore: 5

FAI LA TUA SCELTA

1 Inserimeto
2 Eliminazione
3 Visualizzazione
0 Esci

SCELTA : 3

[3] [4] [5]

FAI LA TUA SCELTA

1 Inserimeto
2 Eliminazione
3 Visualizzazione
0 Esci

SCELTA :

P2_08_04_T

1.[liv.1] Simulare in C l'algoritmo di visita di una *lista lineare* memorizzata mediante un array (statico) di struct, in cui il primo campo contiene le informazioni ed il secondo contiene i link (che in questo caso sono indici alle componenti dell'array) al prossimo nodo. Memorizzando nell'array i dati come mostrato nella figura che segue, l'output del programma consiste nell'elenco di nomi ordinato alfabeticamente.

dati di input: ListaNomi

ListaNomi	
INFO	pnext
Anna	5
Mario	8
Giuseppe	6
Angela	0
Valeria	-1
Fabrizio	7
Marianna	1
Giovanni	2
Patrizia	10
Valentina	4
Sara	9

p_Testa

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define lung 11
//struct con il campo nome delle persone e link
struct persona{
    char nome[10];
    int link; }

//elenco delle persone con un array del tipo della struttura
elenco[lung]={ {"anna",5},{ "mario",8},{ "giuseppe",6},{ "angela",0},
{"valeria",-1},{ "fabrizio",7},{ "marianna",1},{ "giovanni",2},
{"patrizia",10},{ "valentina",4},{ "sara",9}};
typedef struct persona PERSONA;

void list(PERSONA elenco[]);

int main(void)
{
    int i,v=0;
    printf("PERSONA    LINK\n");
    //visualizza i nodi nell'elenco e i corrispettivi link al nodo successivo
    for(i=0;i<lung;i++)
        printf("%s    \t%d\n",elenco[i].nome,elenco[i].link);
    list(elenco);
    return 0; }

void list(PERSONA elenco[])    //visita tramite i link della lista
{
    int j=0,uno=3;
    int *punt=&uno; //puntatore al link di ogni persona
    printf("\n");
    for(j=0;j<lung;j++)
    {
        printf("%s    \t%d\n",elenco[*punt].nome,elenco[*punt].link);
        //ad ogni ciclo ci si collega alla persona successiva tramite il link assegnatogli
        *punt=elenco[*punt].link;
    }
    printf("\n");
}
```

```
C:\Windows\system32\cmd.exe

PERSONA      LINK
anna         5
mario        8
giuseppe     6
angela       0
valeria      -1
fabrizio     7
marianna     1
giovanni     2
patrizia     10
valentina    4
sara         9

angela       0
anna         5
fabrizio     7
giovanni     2
giuseppe     6
marianna     1
mario        8
patrizia     10
sara         9
valentina    4
valeria      -1

Premere un tasto per continuare . . . _
```

P2_08_08_AT

*1.[liv.1] Realizzare in C le funzioni per la gestione delle strutture dati **pila e coda** mediante **lista lineare dinamica e generica** con [rispettivamente senza] nodo sentinella.*

Coda con Sentinella:

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

struct PERSONA {
    char info[20];
    struct PERSONA *p_next;};

void crea_lista(struct PERSONA **);

void elil_testa(struct PERSONA **);

void insl_nodo(struct PERSONA **);

void stampa_lista(struct PERSONA *);

/******
void main()
/******
{
    short menu, numrec=1;
    struct PERSONA *head,*bottom;
    crea_lista(&head);
    bottom=head;
    do
    {
        puts("CODA con sentinella, seleziona:");
        puts("[0] uscita programma");
        puts("[1] Inserimento in coda");
        puts("[2] Eliminazione in testa");

        fflush(stdin);
        scanf("%hd",&menu);
        switch(menu)
        { //inserimento in coda
            case 1:
                insl_nodo(&bottom);
                numrec++;
                stampa_lista(head);
                printf("\nnodi:%d\n",numrec);
                break;
            //eliminazione in testa
            case 2:
                if(numrec>1) //per non eliminare la sentinella
                {
                    elil_testa(&head);
                    numrec--;
                }
                if(numrec==1) //siamo arrivati al nodo sentinella
                    bottom=head;
                stampa_lista(head);
                printf("\nnodi:%d\n",numrec);
                break;
            default: break;
        }
    }
```

```

    } while(menu!=0);
}
//*****
void crea_lista(struct PERSONA **p_head)
//*****
{
    struct PERSONA *testa;
    testa=calloc(1,sizeof(struct PERSONA));
    strcpy(testa->info,"SENTIN");
    *p_head=testa;
}
//*****
void elil_testa(struct PERSONA **p_head)
//*****
{
    //con il nodo sentinella agisco su
    //head->p_next invece di head
    struct PERSONA *ptr;
    if(*p_head!=NULL)
    {
        ptr=(*p_head)->p_next;
        if(ptr->p_next!=NULL)
        {
            (*p_head)->p_next=ptr->p_next;
        }
        else
            (*p_head)->p_next=NULL;
        free(ptr);
    }
}
//*****
void insl_nodo(struct PERSONA **p_bottom)
//*****
{
    //con il nodo sentinella agisco su
    //bottom->p_next invece di bottom
    struct PERSONA *ptr;
    char stringa[20];
    ptr=calloc(1, sizeof(struct PERSONA));
    printf("Inserire una descrizione:");
    scanf("%s",&stringa);
    strcpy(ptr->info,stringa);
    //collego al nuovo il precedente
    (*p_bottom)->p_next=ptr;
    *p_bottom=ptr;
}

```



```

/*****
void stampa_lista(struct PERSONA *p_head)
*****/
{
    struct PERSONA *ptr;
    short cont=0,esito=1;
    ptr=p_head;
    if(ptr->info!=NULL)
    {
        printf("posiz\tdescr\tattuale\tp_next\n");
        do
        {
            cont++;
            printf("%d\t%s\t%d\t%d\n",cont,ptr->info,ptr,ptr->p_next);
            if(ptr->p_next!=NULL)
                ptr=ptr->p_next;
            else
                esito=0;
        }
        while(esito!=0);
        printf("Indirizzo di testa:%d\n",p_head);
        printf("\n");
    }
}

```

```
C:\Windows\system32\cmd.exe
CODA con sentinella, seleziona:
[0] uscita programma
[1] Inserimento in coda
[2] Eliminazione in testa
1
Inserire una descrizione:2
posiz  descr  attuale p_next
1      SENTIN 1842112 1842184
2      2      1842184 0
Indirizzo di testa:1842112

nodi:2
CODA con sentinella, seleziona:
[0] uscita programma
[1] Inserimento in coda
[2] Eliminazione in testa
1
Inserire una descrizione:3
posiz  descr  attuale p_next
1      SENTIN 1842112 1842184
2      2      1842184 1842256
3      3      1842256 0
Indirizzo di testa:1842112

nodi:3
CODA con sentinella, seleziona:
[0] uscita programma
[1] Inserimento in coda
[2] Eliminazione in testa
1
Inserire una descrizione:4
posiz  descr  attuale p_next
1      SENTIN 1842112 1842184
2      2      1842184 1842256
3      3      1842256 1842328
4      4      1842328 0
Indirizzo di testa:1842112

nodi:4
CODA con sentinella, seleziona:
[0] uscita programma
[1] Inserimento in coda
[2] Eliminazione in testa
2
posiz  descr  attuale p_next
1      SENTIN 1842112 1842256
2      3      1842256 1842328
3      4      1842328 0
Indirizzo di testa:1842112

nodi:3
CODA con sentinella, seleziona:
[0] uscita programma
[1] Inserimento in coda
[2] Eliminazione in testa
2
posiz  descr  attuale p_next
```

Coda Senza Sentinella:

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

struct PERSONA {
    char info[20];
    struct PERSONA *p_next;};

struct PERSONA *crea_lista();

void elil_testa(struct PERSONA **);

void insl_nodo(struct PERSONA **);

void insl_testa(struct PERSONA **);

void stampa_lista(struct PERSONA *);

/******
void main()
/******
{
    short menu, numrec=0;
    struct PERSONA *head,*bottom;
    head=crea_lista();
    do
    {
        puts("CODA senza sentinella, seleziona:");
        puts("[0] uscita programma");
        puts("[1] Inserimento in coda");
        puts("[2] Eliminazione in testa");
        fflush(stdin);
        scanf("%hd",&menu);
        switch(menu)
        {
            //inserimento in testa(1 volta) e coda
            case 1:
                if(numrec==0)
                {
                    insl_testa(&head);
                    bottom=head;
                }
                else
                    insl_nodo(&bottom);
                numrec++;
                stampa_lista(head);
                printf("\nnodi:%d\n",numrec);
                break;
            //eliminazione in testa
            case 2:
                elil_testa(&head);
                if(numrec>0)
                    numrec--;
                stampa_lista(head);
                printf("\nnodi:%d\n",numrec);
                break;
            default: break;
        }
    } while(menu!=0);
}
```

```

//*****
struct PERSONA *crea_lista()
//*****
{
    struct PERSONA *testa;
    testa=NULL;
    return testa;
}

//*****
void elil_testa(struct PERSONA **p_head)
//*****
{
    struct PERSONA *ptr;
    if(*p_head!=NULL)
    {
        ptr=*p_head;
        if(ptr->p_next!=NULL)
        {
            *p_head=ptr->p_next;
        }
        else
            *p_head=NULL;
        free(ptr);
    }
}

//*****
void insl_nodo(struct PERSONA **p_bottom)
//*****
{
    struct PERSONA *ptr;
    char stringa[20];
    ptr=calloc(1, sizeof(struct PERSONA));

    printf("Inserire una descrizione:");
    scanf("%s",&stringa);
    strcpy(ptr->info,stringa);

    (*p_bottom)->p_next=ptr;

    *p_bottom=ptr;
}

//*****
void insl_testa(struct PERSONA **p_head)
//*****
{
    struct PERSONA *ptr;
    char stringa[20];
    ptr=calloc(1, sizeof(struct PERSONA));
    printf("Inserire una descrizione:");
    scanf("%s",&stringa);
    strcpy(ptr->info,stringa);
    ptr->p_next=*p_head;
    *p_head=ptr;
}

```

```

/*****
void stampa_lista(struct PERSONA *p_head)
*****/
{
    struct PERSONA *ptr;
    short cont=0,esito=1;
    ptr=p_head;
    if(ptr->info!=NULL)
    {
        //system("cls");
        printf("posiz\tdescr\tattuale\tp_next\n");
        do
        {
            cont++;
            printf("%d\t%s\t%d\t%d\n",cont,ptr->info,ptr,ptr->p_next);
            if(ptr->p_next!=NULL)
                ptr=ptr->p_next;
            else
                esito=0;
        }
        while(esito!=0);
        printf("Indirizzo di testa:%d\n",p_head);
        printf("\n");
    }
}

```

```
C:\Windows\system32\cmd.exe
CODA senza sentinella, seleziona:
[0] uscita programma
[1] Inserimento in coda
[2] Eliminazione in testa
1
Inserire una descrizione:A
posiz  descr  attuale p_next
1      A      1121216 0
Indirizzo di testa:1121216

nodi:1
CODA senza sentinella, seleziona:
[0] uscita programma
[1] Inserimento in coda
[2] Eliminazione in testa
1
Inserire una descrizione:B
posiz  descr  attuale p_next
1      A      1121216 1121288
2      B      1121288 0
Indirizzo di testa:1121216

nodi:2
CODA senza sentinella, seleziona:
[0] uscita programma
[1] Inserimento in coda
[2] Eliminazione in testa
1
Inserire una descrizione:C
posiz  descr  attuale p_next
1      A      1121216 1121288
2      B      1121288 1121360
3      C      1121360 0
Indirizzo di testa:1121216

nodi:3
CODA senza sentinella, seleziona:
[0] uscita programma
[1] Inserimento in coda
[2] Eliminazione in testa
2
posiz  descr  attuale p_next
1      B      1121288 1121360
2      C      1121360 0
Indirizzo di testa:1121288

nodi:2
CODA senza sentinella, seleziona:
[0] uscita programma
[1] Inserimento in coda
[2] Eliminazione in testa
1
Inserire una descrizione:D
posiz  descr  attuale p_next
1      B      1121288 1121360
2      C      1121360 1121216
3      D      1121216 0
```

Pila con Nodo Sentinella:

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

struct PERSONA {
    char info[20];
    struct PERSONA *p_next;};

void crea_lista(struct PERSONA **);

void elil_testa(struct PERSONA **);

void insl_testa(struct PERSONA **);

void stampa_lista(struct PERSONA *);

/**********
void main()
/**********
{
    short menu, numrec=1;
    struct PERSONA *head;
    crea_lista(&head);
    do
    {
        puts("PILA con sentinella, seleziona:");
        puts("[0] uscita programma");
        puts("[1] Inserimento in testa");
        puts("[2] Eliminazione in testa");

        fflush(stdin);
        scanf("%hd",&menu);
        switch(menu)
        {
            //inserimento in testa
            case 1:
                insl_testa(&head);
                numrec++;
                stampa_lista(head);
                printf("\nnodi:%d\n",numrec);
                break;
            //eliminazione in testa
            case 2:
                if(numrec>1) //per non eliminare la sentinella
                {
                    elil_testa(&head);
                    numrec--;
                }
                stampa_lista(head);
                printf("\nnodi:%d\n",numrec);
                break;
            default: break;
        }
    } while(menu!=0);
}

/**********
void crea_lista(struct PERSONA **p_head)
/**********
{
```

```

    struct PERSONA *testa;
    testa=calloc(1,sizeof(struct PERSONA));
    strcpy(testa->info,"SENTIN");
    *p_head=testa;
}
//*****
void elil_testa(struct PERSONA **p_head)
//*****
{
    //con il nodo sentinella agisco su
    //head->p_next invece di head
    struct PERSONA *ptr;
    if(*p_head!=NULL)
    {
        ptr=(*p_head)->p_next;
        if(ptr->p_next!=NULL)
        {
            (*p_head)->p_next=ptr->p_next;
        }
        else
            (*p_head)->p_next=NULL;
        free(ptr);
    }
}
//*****
void insl_testa(struct PERSONA **p_head)
//*****
{
    //con il nodo sentinella agisco su
    //head->p_next invece di head
    struct PERSONA *ptr;
    char stringa[20];
    ptr=calloc(1, sizeof(struct PERSONA));
    printf("Inserire una descrizione:");
    scanf("%s",&stringa);
    strcpy(ptr->info,stringa);
    //collego al nuovo il precedente
    ptr->p_next=(*p_head)->p_next;
    (*p_head)->p_next=ptr;
}
//*****
void stampa_lista(struct PERSONA *p_head)
//*****
{
    struct PERSONA *ptr;
    short cont=0,esito=1;
    ptr=p_head;
    if(ptr->info!=NULL)
    {
        printf("posiz\tdescr\tattuale\tp_next\n");
        do
        {
            cont++;
            printf("%d\t%s\t%d\t%d\n",cont,ptr->info,ptr,ptr->p_next);
            if(ptr->p_next!=NULL)
                ptr=ptr->p_next;
            else
                esito=0;
        }
        while(esito!=0);
        printf("Indirizzo di testa:%d\n",p_head);
        printf("\n");
    }
}

```


C:\Windows\system32\cmd.exe

PILA con sentinella, seleziona:
[0] uscita programma
[1] Inserimento in testa
[2] Eliminazione in testa

1

Inserire una descrizione:A

posiz	descr	attuale	p_next
1	SENTIN	3546048	3546120
2	A	3546120	0

Indirizzo di testa:3546048

nodì:2

PILA con sentinella, seleziona:
[0] uscita programma
[1] Inserimento in testa
[2] Eliminazione in testa

1

Inserire una descrizione:B

posiz	descr	attuale	p_next
1	SENTIN	3546048	3546192
2	B	3546192	3546120
3	A	3546120	0

Indirizzo di testa:3546048

nodì:3

PILA con sentinella, seleziona:
[0] uscita programma
[1] Inserimento in testa
[2] Eliminazione in testa

2

posiz	descr	attuale	p_next
1	SENTIN	3546048	3546120
2	A	3546120	0

Indirizzo di testa:3546048

nodì:2

PILA con sentinella, seleziona:
[0] uscita programma
[1] Inserimento in testa
[2] Eliminazione in testa

1

Inserire una descrizione:C

posiz	descr	attuale	p_next
1	SENTIN	3546048	3546192
2	C	3546192	3546120
3	A	3546120	0

Indirizzo di testa:3546048

Pila senza Nodo Sentinella:

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

struct PERSONA {
    char info[20];
    struct PERSONA *p_next;};

struct PERSONA *crea_lista();

void elil_testa(struct PERSONA **);

void insl_testa(struct PERSONA **);

void stampa_lista(struct PERSONA *);

/******
void main()
/******
{
    short menu, numrec=0;
    struct PERSONA *head;
    head=crea_lista();
    do
    {
        puts("PILA senza sentinella, seleziona:");
        puts("[0] uscita programma");
        puts("[1] Inserimento in testa");
        puts("[2] Eliminazione in testa");

        fflush(stdin);
        scanf("%hd",&menu);
        switch(menu)
        {
            //inserimento in testa
            case 1:
                insl_testa(&head);
                numrec++;
                stampa_lista(head);
                printf("\nnodi:%d\n",numrec);
                break;
            //eliminazione in testa
            case 2:
                elil_testa(&head);
                if(numrec>0)
                    numrec--;
                stampa_lista(head);
                printf("\nnodi:%d\n",numrec);
                break;
            default: break;
        }
    } while(menu!=0);
}
/******
struct PERSONA *crea_lista()
/******
{
    struct PERSONA *testa;
    testa=NULL;
```

```

        return testa;
    }
    /*******
void elil_testa(struct PERSONA **p_head)
    /*******
{
    struct PERSONA *ptr;
    if(*p_head!=NULL)
    {
        ptr=*p_head;
        if(ptr->p_next!=NULL)
        {
            *p_head=ptr->p_next;
        }
        else
            *p_head=NULL;
        free(ptr);
    }
}
    /*******
void insl_testa(struct PERSONA **p_head)
    /*******
{
    struct PERSONA *ptr;
    char stringa[20];
    ptr=calloc(1, sizeof(struct PERSONA));
    printf("Inserire una descrizione:");
    scanf("%s",&stringa);
    strcpy(ptr->info,stringa);
    ptr->p_next=*p_head;
    *p_head=ptr;
}
    /*******
void stampa_lista(struct PERSONA *p_head)
    /*******
{
    struct PERSONA *ptr;
    short cont=0,esito=1;
    ptr=p_head;
    if(ptr->info!=NULL)
    {
        //system("cls");
        printf("posiz\tdescr\tattuale\tp_next\n");
        do
        {
            cont++;
            printf("%d\t%s\t%d\t%d\n",cont,ptr->info,ptr,ptr->p_next);
            if(ptr->p_next!=NULL)
                ptr=ptr->p_next;
            else
                esito=0;
        }
        while(esito!=0);
        printf("Indirizzo di testa:%d\n",p_head);
        printf("\n");
    }
}

```

C:\Windows\system32\cmd.exe

PILA senza sentinella, seleziona:

[0] uscita programma
[1] Inserimento in testa
[2] Eliminazione in testa

1

Inserire una descrizione:A

posiz	descr	attuale	p_next
1	A	7871424	0

Indirizzo di testa:7871424

nodi:1

PILA senza sentinella, seleziona:

[0] uscita programma
[1] Inserimento in testa
[2] Eliminazione in testa

1

Inserire una descrizione:B

posiz	descr	attuale	p_next
1	B	7871496	7871424
2	A	7871424	0

Indirizzo di testa:7871496

nodi:2

PILA senza sentinella, seleziona:

[0] uscita programma
[1] Inserimento in testa
[2] Eliminazione in testa

1

Inserire una descrizione:C

posiz	descr	attuale	p_next
1	C	7871568	7871496
2	B	7871496	7871424
3	A	7871424	0

Indirizzo di testa:7871568

nodi:3

PILA senza sentinella, seleziona:

[0] uscita programma
[1] Inserimento in testa
[2] Eliminazione in testa

2

posiz	descr	attuale	p_next
1	B	7871496	7871424
2	A	7871424	0

Indirizzo di testa:7871496

nodi:2

PILA senza sentinella, seleziona:

[0] uscita programma
[1] Inserimento in testa
[2] Eliminazione in testa

1

Inserire una descrizione:D

posiz	descr	attuale	p_next
1	D	7871568	7871496
2	B	7871496	7871424
3	A	7871424	0

P2_09_03_T

1.[liv.1] Scrivere function C per la visita (preorder, inorder e postorder) di un albero binario rappresentato mediante array.

```
#include <stdio.h>
#define n 16

void preorder(short albero[],short i,short k);
void inorder(short albero[],short i,short k);
void postorder(short albero[],short i,short k);

int main (void){
    //albero tramite array 1D
    short albero[n]={0,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14};
    short i=0,k=0,scelta=0;
    printf("Metodo: \nPreorder=1\nInorder=2\nPostorder=3\nEsci=0\n");
    scanf("%d",&scelta);
    getchar();
    switch(scelta){
        case 1: preorder(albero,i,k);break;
        case 2: inorder(albero,i,k);break;
        case 3: postorder(albero,i,k);break;
    }
    printf("\n\nArray ....\n");
    for (i=1;i<n;i++)
        printf("Nodo con valore %d\n",albero[i]);
    return 0;
}

// Analizza Radice - figlio Sx - figlio Dx

void preorder(short albero[n],short i,short k){
    printf("\n\nElemento\tPadre\tFiglio SX\tFiglio DX\n");
    printf("-----\n");
    for(i=1;i<n;i++)
    {
        //valore attuale al passo i-esimo
        printf("%d",albero[i]);
        //calcolo del padre in base alla formula sottostante
        k=i/2;
        if(i!=1){printf("\t%d",albero[k]);}
        if(i==1){printf("\t%d",albero[i]);}
        //calcolo del figlio sx in base alla formula sottostante
        k=2*i;
        if(k<n){printf("\t\t%d",albero[k]);}
        if(k>=n){printf("\t\tND");}
        //calcolo del figlio dx in base alla formula sottostante
        k=2*i+1;
        if(k<n){printf("\t\t\t%d\n",albero[k]);}
        if(k>=n){printf("\t\t\tND\n");}
        printf("\n\n");
    }
}
```

```

/*Foglia SX - RADICE - Foglia DX
viene calcolato con k_s la foglia sinistre
con k_r la radice della foglia sinistra
con k_d la foglia destre del subalbero
con k viene calcolato l'indice di ogni radice del sub albero */

```

```

void inorder(short albero[],short i,short k){
    short k_r=0,k_s=0,k_d=0,n_c=n;
    printf("\n\n");
    do{
        k_s=(n_c/2); printf("Visualizzazione nodo Inorder %d\n",albero[k_s]);
        k_r=(k_s/2); printf("Visualizzazione nodo Inorder %d\n",albero[k_r]);
        k_d=(k_s+1); printf("Visualizzazione nodo Inorder %d\n",albero[k_d]);
        n_c=(k_d+1)*2;
        if(k>=2){k=0;}
        if(k_d<n-1){
            k_r=(k_r/2)-k; printf("Visualizzazione nodo Inorder %d\n",albero[k_r]);}
        k++;
    }while(k_d<(n-1));
}

```

```

/*Foglia SX - Foglia DX - RADICE
viene calcolato con k_s la foglia sinistre
con k_r la radice della foglia sinistra
con k_d la foglia destre del subalbero
con k viene calcolato l'indice di ogni radice del sub albero */

```

```

void postorder(short albero[],short i,short k){
    short k_r=0,k_s=0,k_d=0,n_c=n,flag=0;
    k=-1; printf("\n\n");
    do{
        //permette di controllare anche la radice dell albero
        //senza avere valori inesistenti

        if(k<=0 && flag<=3){
            k_s=(n_c/2); printf("Visualizzazione nodo Postorder %d\n",albero[k_s]);
            k_d=(k_s+1); printf("Visualizzazione nodo Postorder %d\n",albero[k_d]);
            k_r=((k_s)/2); printf("Visualizzazione nodo Postorder %d\n",albero[k_r]);
            n_c=(k_d+1)*2;}
        k++;
        //calcola la radice del sub sub albero
        //permette di controllare anche la radice dell albero
        //senza avere valori inesistenti usando la clausola flag==4
        if(k==1 || flag==4){
            k_r=((k_r-1)/2); printf("Visualizzazione nodo Postorder %d\n",albero[k_r]); k=-1;}
        flag++; //controllo sul termine del ciclo
        // 4 e il livello dell' albero +1 per permettere la stampa della radice dell albero
    }while(flag<(n/4)+1);
}

```

PREORDER:

```
C:\Windows\system32\cmd.exe
Metodo:
Preorder=1
Inorder=2
Postorder=3
Esci=0
1

Elemento      Padre      Figlio SX      Figlio DX
-----
0      0      1      2
1      0      3      4
2      0      5      6
3      1      7      8
4      1      9      10
5      2      11      12
6      2      13      14
7      3      ND      ND
8      3      ND      ND
9      4      ND      ND
10     4      ND      ND
11     5      ND      ND
12     5      ND      ND
13     6      ND      ND
14     6      ND      ND

Array ....
Nodo con valore 0
Nodo con valore 1
Nodo con valore 2
Nodo con valore 3
Nodo con valore 4
Nodo con valore 5
Nodo con valore 6
Nodo con valore 7
Nodo con valore 8
Nodo con valore 9
Nodo con valore 10
Nodo con valore 11
Nodo con valore 12
Nodo con valore 13
Nodo con valore 14
Premere un tasto per continuare . . . _
```

INORDER:

C:\Windows\system32\cmd.exe

Metodo:

Preorder=1

Inorder=2

Postorder=3

Esci=0

2

Visualizzazione nodo Inorder 7
Visualizzazione nodo Inorder 3
Visualizzazione nodo Inorder 8
Visualizzazione nodo Inorder 1
Visualizzazione nodo Inorder 9
Visualizzazione nodo Inorder 4
Visualizzazione nodo Inorder 10
Visualizzazione nodo Inorder 0
Visualizzazione nodo Inorder 11
Visualizzazione nodo Inorder 5
Visualizzazione nodo Inorder 12
Visualizzazione nodo Inorder 2
Visualizzazione nodo Inorder 13
Visualizzazione nodo Inorder 6
Visualizzazione nodo Inorder 14

Array

Nodo con valore 0

Nodo con valore 1

Nodo con valore 2

Nodo con valore 3

Nodo con valore 4

Nodo con valore 5

Nodo con valore 6

Nodo con valore 7

Nodo con valore 8

Nodo con valore 9

Nodo con valore 10

Nodo con valore 11

Nodo con valore 12

Nodo con valore 13

Nodo con valore 14

Premere un tasto per continuare . . .

POSTORDER:

```
C:\Windows\system32\cmd.exe
Metodo:
Preorder=1
Inorder=2
Postorder=3
Esci=0
3

Visualizzazione nodo Postorder 7
Visualizzazione nodo Postorder 8
Visualizzazione nodo Postorder 3
Visualizzazione nodo Postorder 9
Visualizzazione nodo Postorder 10
Visualizzazione nodo Postorder 4
Visualizzazione nodo Postorder 1
Visualizzazione nodo Postorder 11
Visualizzazione nodo Postorder 12
Visualizzazione nodo Postorder 5
Visualizzazione nodo Postorder 13
Visualizzazione nodo Postorder 14
Visualizzazione nodo Postorder 6
Visualizzazione nodo Postorder 2
Visualizzazione nodo Postorder 0

Array ....
Nodo con valore 0
Nodo con valore 1
Nodo con valore 2
Nodo con valore 3
Nodo con valore 4
Nodo con valore 5
Nodo con valore 6
Nodo con valore 7
Nodo con valore 8
Nodo con valore 9
Nodo con valore 10
Nodo con valore 11
Nodo con valore 12
Nodo con valore 13
Nodo con valore 14
Premere un tasto per continuare . . .
```

P2_09_06_T

1.[liv.2] Scrivere function C iterativa per la costruzione di un **heap** rappresentato mediante array.

```
#include <stdio.h>
#define n 7
void inorder(short albero[],short i,short k,short liv);

int main (void){
    //albero a 3 livelli
    short albero[n]={ 13,5,84,4,8,66,77};
    short i,k=0,liv=3; // e il numeri dei livelli dell albero
    printf("Albero non configurato in Heap\n\n");
    for(i=0;i<n;i++)
        printf("Nodo dell albero e %d\n",albero[i]);
    inorder(albero,i,k,liv);
    printf("\n\nAlbero configurato in Heap\n\n");
    for(i=0;i<n;i++)
        printf("Nodo dell albero e %d\n",albero[i]);
    printf("\n\n");
    return 0;}

////////////////////////////////////

void inorder(short albero[],short i,short k,short liv){
    //k_r=RADICE k_s=foglia sinistre k_d=foglia destra
    // flag e liv sono variabili che controllano il termine dei cicli
    //basandosi sul numero di livelli dell albero
    short k_r=0,k_s=0,k_d=0,n_c=n,temp=0,flag=0;
    for(k=0;k<=liv;k++){
        do{
            if(k_s==n-2)
                n_c=(k_s-(liv));
            //formula tramite array pe calcolo radice e foglie Dx e Sx
            k_s=(n_c/2);
            k_r=(k_s/2);
            k_d=(k_s+1);
            // controllo tra foglie e radice x il controllo del max e min
            if(albero[k_r]<albero[k_s])
                { temp=albero[k_s];
                  albero[k_s]=albero[k_r];
                  albero[k_r]=temp;}
            if(albero[k_r]<albero[k_d])
                { temp=albero[k_d];
                  albero[k_d]=albero[k_r];
                  albero[k_r]=temp;}
            //aggiornamento nuovo valore max
            n_c=(k_d+1)*2; flag++;
            // viene ripetuto il ciclo tante volte quante sono i livelli
            //per aggiornare i valori nel caso peggiore
        } while(flag<liv);}
    }

////////////////////////////////////
```

C:\Windows\system32\cmd.exe

Albero non configurato in Heap

Nodo dell1 albero e 13
Nodo dell1 albero e 5
Nodo dell1 albero e 84
Nodo dell1 albero e 4
Nodo dell1 albero e 8
Nodo dell1 albero e 66
Nodo dell1 albero e 77

Albero configurato in Heap

Nodo dell albero e 84
Nodo dell albero e 8
Nodo dell albero e 77
Nodo dell albero e 4
Nodo dell albero e 5
Nodo dell albero e 13
Nodo dell albero e 66

Premere un tasto per continuare . . .

P2_10_02_T

1.[liv.1] Scrivere function C per la costruzione di un grafo non orientato mediante matrice di adiacenze: in input per ogni nodo sono specificati quegli adiacenti. Scegliendo in input un nodo, scrivere function C che restituisca il suo grado.

```
#include <stdio.h>
#define max 50
void grado(short gradi[max],short count,short k,short n);
void matrice(short gradi_mat[][max],short j,short i,short n);
void stampa( short gradi_mat[][max],short j,short i,short n);

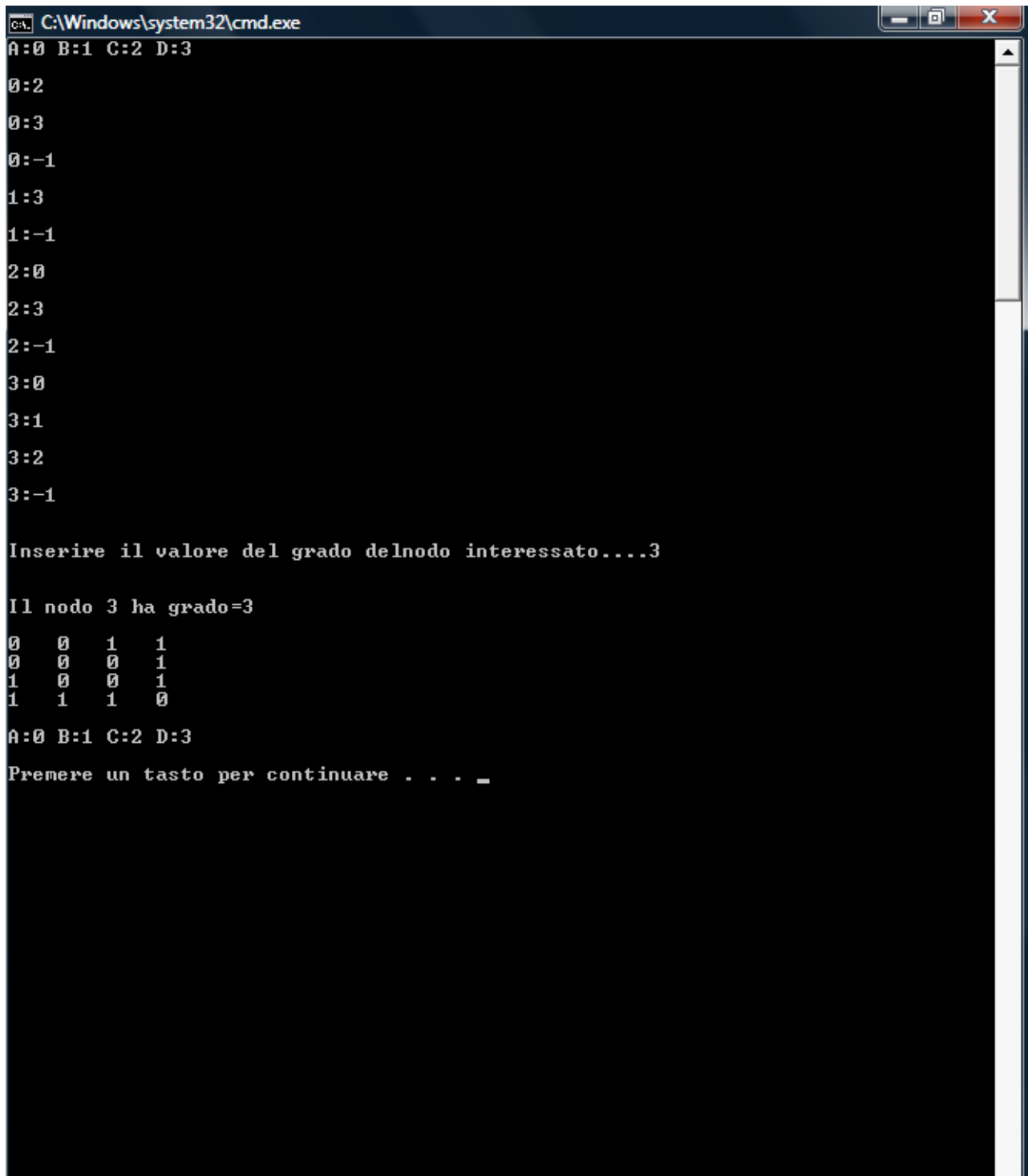
int main (void){
    short gradi_mat[max][max];
    short i=0,j=0,n=4;
    printf("A:0 B:1 C:2 D:3\n\n");
    matrice(gradi_mat,j,i,n);
    stampa(gradi_mat,j,i,n);
    return 0;
}

void matrice(short gradi_mat[max][max],short j,short i,short n){
    short gradi[4],k=0,count=0;
    //IMPOSTA TUTTI I VALORI DELLA MATRICE A 0
    for (i=0;i<n;i++){
        for(j=0;j<n;j++){
            gradi_mat[i][j]=0;
        }
    }
    //IMPOSTA I VALORI DEI NODI CORRELATI AL NODO IN QUESTIONE A 1
    for(i=0;i<n;i++){
        do{
            printf("%d:",i); scanf("%d",&j);printf("\n");
            if(j!=-1)
            {
                gradi_mat[i][j]=1;
                count++;
            }
        } while(j!=-1);
        //TERMINA DI CORRELARE NODI AL NODO PRINCIPALE IMPOSTANDO -1
        gradi[k]=count; k++;count=0;//conta i gradi dei nodi
    }
    //FUNZIONE CHE CALCOLA IL GRADO DEI NODI INSERITI DA SCANF
    grado(gradi,count,k,n);
}

//FUNZIONE CHE CALCOLA IL GRADO del nodo inserito da scanf
void grado(short gradi[max],short count,short k,short n){
    short massimo=0;
    printf("\nInserire il valore del grado del nodo interessato....");
    scanf("%d",&k);
    printf("\nIl nodo %d ha grado=%d\n",k,gradi[k]);
}
```

//FUNZIONE STAMPA MATRICE

```
void stampa( short gradi_mat[][max],short j,short i,short n){  
    for (i=0;i<n;i++){  
        for(j=0;j<n;j++){  
            {  
                printf("%d  ",gradi_mat[i][j]);  
            }  
            printf("\n");  
        }  
        printf("\nA:0 B:1 C:2 D:3\n\n");  
    }  
}
```



```
C:\Windows\system32\cmd.exe  
A:0 B:1 C:2 D:3  
0:2  
0:3  
0:-1  
1:3  
1:-1  
2:0  
2:3  
2:-1  
3:0  
3:1  
3:2  
3:-1  
  
Inserire il valore del grado delnodo interessato....3  
  
Il nodo 3 ha grado=3  
0 0 1 1  
0 0 0 1  
1 0 0 1  
1 1 1 0  
  
A:0 B:1 C:2 D:3  
Premere un tasto per continuare . . . _
```

2.[liv.1] Scrivere function C per la costruzione di un grafo orientato mediante matrice di adiacenze: in input per ogni nodo sono specificati quelli raggiungibili. Scegliendo in input un nodo, scrivere function C che restituisca il numero degli archi uscenti e quello degli archi entranti.

```
#include <stdio.h>
#define max 50
void grado(short [],short ,short ,short ,short [][max]);
void matrice(short [][max],short ,short ,short );
void stampa( short [][max],short ,short ,short );

int main (void){
    short gradi_mat[max][max];
    short i=0,j=0,n=4;
    printf("A:0 B:1 C:2 D:3\n\n");
    matrice(gradi_mat,j,i,n);
    stampa(gradi_mat,j,i,n);
    return 0;
}
////////////////////////////////////

// Inserimento nodi del Grafo nella matrice
void matrice(short gradi_mat[max][max],short j,short i,short n){
    short gradi[4],k=0,count=0;
    //IMPOSTA TUTTI I VALORI DELLA MATRICE A 0
    for (i=0;i<n;i++){
        for(j=0;j<n;j++){
            {
                gradi_mat[i][j]=0;
            }
        }
    }
    //IMPOSTA I VALORI DEI NODI CORRELATI AL NODO IN QUESTIONE A 1
    for(i=0;i<n;i++){
        do{
            printf("%d:",i); scanf("%d",&j);printf("\n");
            if(j!=-1)
            {
                gradi_mat[i][j]=1;
                count++;
            }
        } while(j!=-1);
        gradi[k]=count; k++; count=0;//conta i gradi dei nodi
    }
    //FUNZIONE CHE CALCOLA IL GRADO DEI NODI INSERITI DA SCANF
    grado(gradi,count,k,n,gradi_mat);
}
////////////////////////////////////

//FUNZIONE CHE CALCOLA IL GRADO del nodo inserito da scanf
void grado(short gradi[max],short count,short k,short n,short gradi_mat[][max]){
    short i;
    printf("\nInserire il valore del grado del nodo interessato....");
    scanf("%d",&k);
    printf("\nIl nodo %d ha (quanti nodi si collega)grado=%d\n",k,gradi[k]);
    //Calcola gli archi che vanno verso il nodo K
    for(i=0;i<n;i++){
        if(gradi_mat[i][k]==1)
            count++;
    }
    printf("\nIl nodo %d ha (quanti nodi gli si collegano)grado=%d\n",k,count);
}
```

```

////////////////////////////////////

//FUNZIONE STAMPA MATRICE
void stampa( short gradi_mat[][max],short j,short i,short n){
    for (i=0;i<n;i++){
        for(j=0;j<n;j++)
        {
            printf("%d ",gradi_mat[i][j]);
        }
        printf("\n");
    }
    printf("\nA:0 B:1 C:2 D:3\n\n");
}

```

```

C:\Windows\system32\cmd.exe
A:0 B:1 C:2 D:3
0:3
0:-1
1:-1
2:0
2:-1
3:1
3:2
3:-1

Inserire il valore del grado del nodo interessato....2

Il nodo 2 ha <quanti nodi si collega>grado=1

Il nodo 2 ha <quanti nodi gli si collegano>grado=1
0  0  0  1
0  0  0  0
1  0  0  0
0  1  1  0

A:0 B:1 C:2 D:3
Premere un tasto per continuare . . . _

```

P2_11_01_T

1.[liv.1] Scrivere delle function C (rispettivamente iterativa e ricorsiva) per calcolare (con ricorsione sia lineare sia binaria) la somma delle componenti di un array.

```
#include <stdio.h>
int somma_ric_lineare(int array[], int dim);
int somma_ric_binaria(int array[], int dim);

void main()
{
    int array[]={0,1,2,3,4,5,6,7,8,9};
    int n=10, i;
    printf("Array:\n");
    for(i=0;i<n;i++)
        printf("%d\t",array[i]);
    printf("\n\nSomma ricorsiva lineare: %d",somma_ric_lineare(array,n));
    printf("\n\nSomma ricorsiva binaria: %d\n\n",somma_ric_binaria(array,n));
}

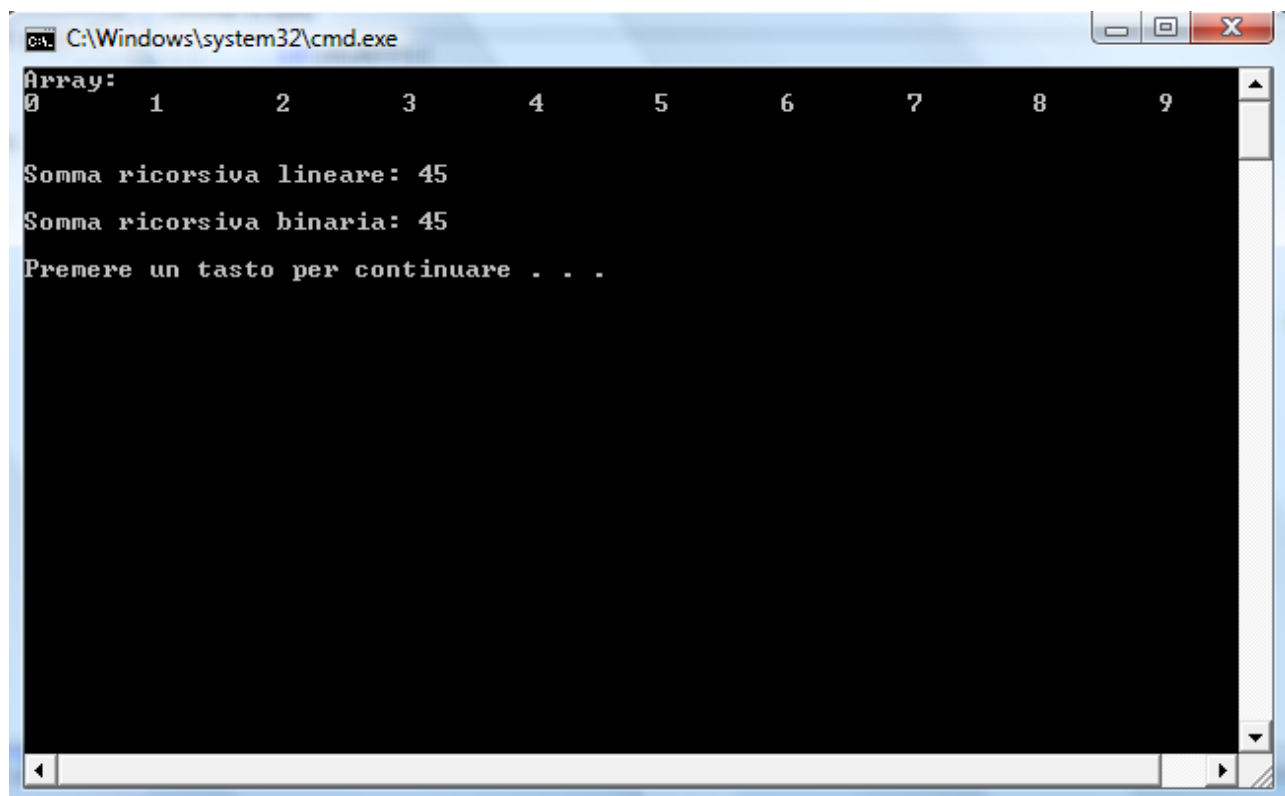
/* se la dimensione dell array (dim) e 1
allora ritorna array[0] somma dei valori*/
/* In caso che dim on e 1 allora riduci l'array e somma il
valore attuale passando il valore dell array accorciato*/

int somma_ric_lineare(int array[], int dim)
{
    if(dim==1)
        return array[dim-1];
    else
        return array[dim-1]+somma_ric_lineare(array,dim-1);
}

/* se la dimensione e <=1 allora la somma e terminata
se dim (dimensione array) e maggiore di 2 allora ritorna
alla funzione il valore sommato precedentemente + il valore attuale e riduce l'array*/

int somma_ric_binaria(int array[], int dim)
{
    if(dim<=1)
        return array[0];

    if(dim>=2)
        return array[dim-1]+somma_ric_binaria(array,dim-1);
    else
        return somma_ric_binaria(array,dim-1);
}
```

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Windows\system32\cmd.exe'. The command prompt has a black background with white text. It displays an array of numbers from 0 to 9, followed by two lines of recursive sum results, and a prompt to press a key to continue.

```
C:\Windows\system32\cmd.exe  
Array:  
0      1      2      3      4      5      6      7      8      9  
  
Somma ricorsiva lineare: 45  
Somma ricorsiva binaria: 45  
Premere un tasto per continuare . . .
```

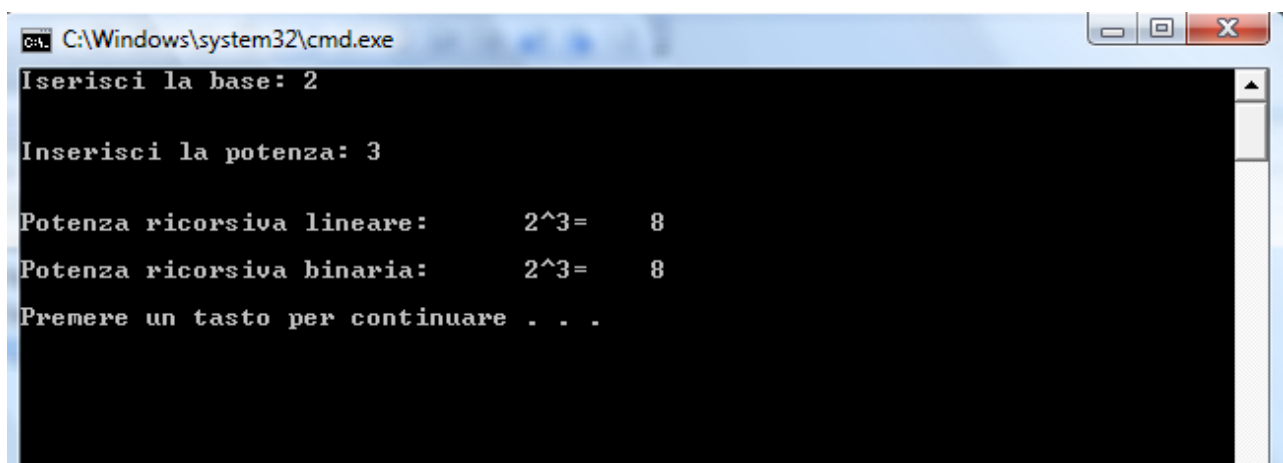
2.[liv.1] Scrivere delle *function C* (rispettivamente iterativa e ricorsiva) per calcolare (con ricorsione sia lineare sia binaria) la potenza intera x^n di un numero reale.

```
#include <stdio.h>
int potenza_ric_lin(int base, int potenza);
int potenza_ric_bin(int base, int potenza);

// X e la base inserita da Input
// N e il valore della potenza inserita da Input (ESPONENTE)
void main()
{
    int x,n;
    printf("Iserisci la base: ");
    scanf("%d",&x);
    printf("\n\nIserisci la potenza: ");
    scanf("%d",&n);
    printf("\n\nPotenza ricorsiva lineare:\t%d^%d=\t%d\n\n",x,n,potenza_ric_lin(x,n));
    printf("Potenza ricorsiva binaria:\t%d^%d=\t%d\n\n",x,n,potenza_ric_bin(x,n));
}

//in caso che Potenza==N e uguale a 1 valore trovato
//in caso contrario la base viene moltiplicata
//per il valore moltiplicato precedentemente
int potenza_ric_lin(int base, int potenza)
{
    if(potenza==1)
        return base;
    else
        return base*potenza_ric_lin(base,potenza-1);
}

//in caso che Potenza==N e uguale a 1 valore trovato
//in caso contrario la base viene moltiplicata
//per il valore moltiplicato precedentemente
int potenza_ric_bin(int base, int potenza)
{
    if(potenza==1)
        return base;
    else if(potenza>=2)
        return base*potenza_ric_bin(base,potenza-1);
    else
        return potenza_ric_bin(base,potenza-1);
}
```



```
C:\Windows\system32\cmd.exe
Iserisci la base: 2

Iserisci la potenza: 3

Potenza ricorsiva lineare:      2^3=      8
Potenza ricorsiva binaria:      2^3=      8
Premere un tasto per continuare . . .
```

P2_12_01_T

1.[liv.1] Scrivere due function C (rispettivamente iterativa e ricorsiva) per implementare l'algoritmo **Selection Sort** su un array di struttura, sia mediante scambi reali sia mediante scambi virtuali.

(SCAMBI REALI)

```
#include<stdio.h>
#define MAX_NUM 10

struct selection_sort{
    char info;
}array[MAX_NUM];
struct selection_sort array2[MAX_NUM];

void max_val_ind(struct selection_sort a[], int n, struct selection_sort *max_a, int *ind_max);
void ord_sel_max_iter(struct selection_sort a[], int n);
void scambiare_c(struct selection_sort *c1, struct selection_sort *c2);
void ord_sel_max_ric(struct selection_sort a[], int n);

void main()
{
    int i;
    printf("Inserire i caratteri da ordinare\n");
    for(i=0;i<MAX_NUM;i++)
    {
        printf("\n%do carattere: ",i+1);
        scanf("%c",&array[i].info);
        array2[i].info=array[i].info;
        fflush(stdin);
    }
    printf("\nArray non ordinato: ");
    for(i=0;i<MAX_NUM;i++)
        printf("%c ",array[i].info);
    ord_sel_max_iter(array,MAX_NUM);//chiamata alla function per iterazione
    printf("\n\nArray ordinato(MODO ITERATIVO):  ");
    for(i=0;i<MAX_NUM;i++)
        printf("%c ",array[i].info);
    printf("\n\n");
    ord_sel_max_ric(array2,MAX_NUM);//chiamata alla function per la ricorsione
    printf("\n\nArray ordinato(MODO RICORSIVO):  ");
    for(i=0;i<MAX_NUM;i++)
        printf("%c ",array2[i].info);
    printf("\n\n");
}

void max_val_ind(struct selection_sort a[], int n, struct selection_sort *max_a, int *ind_max)
{
    int i;
    *max_a=a[0];
    *ind_max=0;
    for(i=0;i<n;i++)
    {
        if(a[i].info>*max_a->info)
        {
            *max_a=a[i];
            *ind_max=i;
        }
    }
}
```

```

void ord_sel_max_iter(struct selection_sort a[], int n)
{
    struct selection_sort max_a;
    int ind_max,i;
    for(i=n-1;i>=0;i--)
    {
        max_val_ind(&a[0],i+1,&max_a,&ind_max);//determina il massimo
        scambiare_c(&a[i],&a[ind_max]);//scambialo con la posizione i-esima che al primo passo è l'ultima
        componente dell'array
    }
}

void scambiare_c(struct selection_sort *c1, struct selection_sort *c2)
{
    struct selection_sort temp;
    temp=*c1;
    *c1=*c2;//scambio di elementi tramite variabile di appoggio
    *c2=temp;
}

void ord_sel_max_ric(struct selection_sort a[], int n)//ricorsiva
{
    struct selection_sort max_a;
    int ind_max;
    if(n==2)
    {
        max_val_ind(&a[0],n,&max_a,&ind_max);
        scambiare_c(&a[n-1],&a[ind_max]);
    }
    else
    {
        max_val_ind(&a[0],n,&max_a,&ind_max);
        scambiare_c(&a[n-1],&a[ind_max]);
        ord_sel_max_ric(a,n-1);//restituisce l'array ordinato
    }
}

```

```
C:\Windows\system32\cmd.exe
Inserire i caratteri da ordinare
1o carattere: 10
2o carattere: 9
3o carattere: 8
4o carattere: 7
5o carattere: 6
6o carattere: 5
7o carattere: 4
8o carattere: 3
9o carattere: 2
10o carattere: 1
Array non ordinato: 1 9 8 7 6 5 4 3 2 1
Array ordinato(MODO ITERATIVO): 1 1 2 3 4 5 6 7 8 9
Array ordinato(MODO RICORSIVO): 1 1 2 3 4 5 6 7 8 9
Premere un tasto per continuare . . .
```

(SCAMBI VIRTUALI)

```
#include<stdio.h>
#define MAX_NUM 10

struct selection_sort{
    char info;
}array[MAX_NUM];
struct selection_sort array2[MAX_NUM];
struct selection_sort *array_punt[MAX_NUM]; //l'array di puntatori permette gli scambi virtuali

void max_val_ind(struct selection_sort a[], int n, struct selection_sort *max_a, int *ind_max);
void ord_sel_max_iter(struct selection_sort a[], int n);
void assegna_punt(int ind1, int ind2);
void ord_sel_max_ric(struct selection_sort a[], int n);

void main()
{
    int i;
    printf("Inserire i caratteri da ordinare\n");
    for(i=0;i<MAX_NUM;i++)
    {
        printf("\n%do carattere: ",i+1);
        scanf("%c",&array[i].info);
        array2[i].info=array[i].info;
        fflush(stdin);
    }
    printf("\nArray non ordinato: ");
    for(i=0;i<MAX_NUM;i++)
        printf("%c ",array[i].info);
    ord_sel_max_iter(array,MAX_NUM); //chiamata alla function di iterazione
    printf("\n\nArray ordinato(METODO ITERATIVO):  ");
```

```

    for(i=0;i<MAX_NUM;i++)
        printf("%c ",array_punt[i]->info);
    printf("\n\n");
    for(i=0;i<MAX_NUM;i++)
        array_punt[i]=&array2[i];
    ord_sel_max_ric(array2,MAX_NUM);//chiamata alla function di ricorsione
    printf("\n\nArray ordinato(METODO RICORSIVO):  ");
    for(i=0;i<MAX_NUM;i++)
        printf("%c ",array_punt[i]->info);
    printf("\n\n");
}

```

```

void max_val_ind(struct selection_sort a[], int n, struct selection_sort *max_a, int *ind_max)
{
    int i;
    max_a=array_punt[0];
    *ind_max=0;
    for(i=1;i<n;i++)
    {
        if((array_punt[i]->info)>(max_a->info))//si cerca il massimo
        {
            max_a=array_punt[i];//assegna il massimo
            *ind_max=i;//assegna il suo indice
        }
    }
}

```

//restituisce array di puntatori a struct che puntano all'array di struct non ordinato (il primo elemento dell'array //di puntatori punta al minimo elemento dell'array di struct, il secondo a quello successivo...e l'ultimo al massimo

```

void ord_sel_max_iter(struct selection_sort a[], int n)
{
    struct selection_sort max_a;
    int ind_max,i;
    for(i=0;i<n;i++)
        array_punt[i]=&a[i];
    for(i=n-1;i>=0;i--)
    {
        max_val_ind(a,i+1,&max_a,&ind_max);
        assegna_punt(i,ind_max);
    }
}

```

```

void assegna_punt(int ind1, int ind2)
{
    struct selection_sort *temp;
    temp=array_punt[ind1];
    array_punt[ind1]=array_punt[ind2];//scambio indice di puntatori tramite variabile d'appoggio
    array_punt[ind2]=temp;
}

```

//restituisce array di puntatori a struct che puntano all'array di struct non ordinato (il primo elemento dell'array //di puntatori punta al minimo elemento dell'array di struct, il secondo a quello successivo...e l'ultimo al massimo

```

void ord_sel_max_ric(struct selection_sort a[], int n)
{
    struct selection_sort max_a;
    int ind_max;
    if(n==2)
    {
        max_val_ind(a,n,&max_a,&ind_max);
        assegna_punt(n-1,ind_max);
    }
    else
    {
        max_val_ind(a,n,&max_a,&ind_max);
        assegna_punt(n-1,ind_max);
        ord_sel_max_ric(a,n-1);
    }
}

```

```

    }
}

```

```

C:\Windows\system32\cmd.exe
1o carattere: 10
2o carattere: 1
3o carattere: 9
4o carattere: 8
5o carattere: 7
6o carattere: 6
7o carattere: 5
8o carattere: 4
9o carattere: 3
10o carattere: 2
Array non ordinato: 1 1 9 8 7 6 5 4 3 2
Array ordinato(METODO ITERATIVO): 1 1 2 3 4 5 6 7 8 9
Array ordinato(METODO RICORSIVO): 1 1 2 3 4 5 6 7 8 9
Premere un tasto per continuare . . .

```

2.*[liv.1]* Scrivere una function C per implementare l'**algoritmo Bubble Sort** su un array di struttura, sia mediante scambi reali sia mediante scambi virtuali.

(SCAMBI REALI)

```

#include<stdio.h>
#define MAX_NUM 10

struct bubble_sort{
    char info;
}array[MAX_NUM];

void ordina_struct_bubble(struct bubble_sort a[], int n);
void scambia(struct bubble_sort *c1, struct bubble_sort *c2);
void main()
{
    int i;
    printf("\t\t\tInserisci caratteri da ordinare\n");
    for(i=0;i<MAX_NUM;i++)
    {
        printf("\n%do elemento: ",i+1);
        scanf("%c",&array[i].info);
        fflush(stdin);
    }
    printf("\n\nArray non ordinato:\n");
    for(i=0;i<MAX_NUM;i++)
        printf("%c ",array[i].info);
    ordina_struct_bubble(array,MAX_NUM);
    printf("\n\nArray ordinato:\n");
    for(i=0;i<MAX_NUM;i++)
        printf("%c ",array[i].info);
    printf("\n\n");
}

```

```

//input: array non ordinato, dimensione array
//output: array ordinato
void ordina_struct_bubble(struct bubble_sort a[], int n)
{
    int i,esc=0;
    while(esc==0)
    {
        esc=1;
        for(i=0;i<n-1;i++)
        {
            if(a[i].info>a[i+1].info)
            {
                scambia(&a[i],&a[i+1]);//se elemento i-esimo maggiore dell'elemento i+1-esimo
                esc=0;
            }
        }
    }
}

//input: elemento i_esimo, elemento i+1-esimo
//output: elementi scambiati
void scambia(struct bubble_sort *c1, struct bubble_sort *c2)
{
    struct bubble_sort temp;
    temp=*c1;
    *c1=*c2;
    *c2=temp;
}

```

```

C:\Windows\system32\cmd.exe
Inserisci caratteri da ordinare

1o elemento: 10
2o elemento: 9
3o elemento: 8
4o elemento: 7
5o elemento: 6
6o elemento: 5
7o elemento: 1
8o elemento: 2
9o elemento: 3
10o elemento: 4

Array non ordinato:
1 9 8 7 6 5 1 2 3 4

Array ordinato:
1 1 2 3 4 5 6 7 8 9

Premere un tasto per continuare . . . _

```


(SCAMBI VIRTUALI)

```
#include<stdio.h>
#define MAX_NUM 10

struct bubble_sort{
    char info;
}array[MAX_NUM];

void ordina_struct_bubble(struct bubble_sort a[], int n);
void scambia(struct bubble_sort *c1, struct bubble_sort *c2);

void main()
{
    int i;
    printf("\t\t\tInserisci caratteri da ordinare\n");
    for(i=0;i<MAX_NUM;i++)
    {
        printf("\n%do elemento: ",i+1);
        scanf("%c",&array[i].info);
        fflush(stdin);
    }
    printf("\n\nArray non ordinato:\n");
    for(i=0;i<MAX_NUM;i++)
        printf("%c ",array[i].info);
    ordina_struct_bubble(array,MAX_NUM);
    printf("\n\nArray ordinato:\n");
    for(i=0;i<MAX_NUM;i++)
        printf("%c ",array[i].info);
    printf("\n\n");
}
```

```

//input: array non ordinato, dimensione array
//output: array ordinato
void ordina_struct_bubble(struct bubble_sort a[], int n)
{
    int i,esc=0;
    while(esc==0)
    {
        esc=1;
        for(i=0;i<n-1;i++)
        {
            if(a[i].info>a[i+1].info)
            {
                scambia(&a[i],&a[i+1]);//se elemento i-esimo maggiore dell'elemento i+1-esimo
                esc=0;
            }
        }
    }
}

//input: elemento i_esimo, elemento i+1-esimo
//output: elementi scambiati
void scambia(struct bubble_sort *c1, struct bubble_sort *c2)
{
    struct bubble_sort temp;
    temp=*c1;
    *c1=*c2;
    *c2=temp;
}

```

```

C:\Windows\system32\cmd.exe
3o elemento: 8
4o elemento: 7
5o elemento: 6
6o elemento: 6
7o elemento: 5
8o elemento: 4
9o elemento: 3
10o elemento: 2
11o elemento: 1
12o elemento: \

Array non ordinato:
1 9 8 7 6 6 5 4 3 2 1 \

Array ordinato:
1 1 2 3 4 5 6 6 7 8 9 \

Premere un tasto per continuare . . .

```

3.*[liv.1]* Scrivere una function C per implementare l'*algoritmo Insertion Sort* su un array di struttura.

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

```
#define MAX_NUM 10
```

```
struct insertion_sort{  
    char info;  
}array[MAX_NUM];
```

```
void ordina_struct_insertion(struct insertion_sort a[], int n);
```

```
void main()  
{  
    int i;  
    printf("\t\t\tInserisci caratteri da ordinare\n");  
    for(i=0;i<MAX_NUM;i++)  
    {  
        printf("\n%do elemento: ",i+1);  
        scanf("%c",&array[i].info);  
        fflush(stdin);  
    }  
    printf("\n\nArray non ordinato:\n");  
    for(i=0;i<MAX_NUM;i++)  
        printf("%c ",array[i].info);  
    ordina_struct_insertion(array,MAX_NUM);  
    printf("\n\nArray ordinato:\n");  
    for(i=0;i<MAX_NUM;i++)  
        printf("%c ",array[i].info);  
    printf("\n\n");  
}
```

```

}

//input: struct non ordinata, dimensione struct
//output: struct ordinata
void ordina_struct_insertion(struct insertion_sort a[], int n)
{
    int i=0, j=0;
    struct insertion_sort temp;
    for(i=1; i<n; i++)
    {
        temp=a[i];
        j=i-1;
        while(j>=0 && a[j].info>temp.info)//finchè l'elemento i-esimo > elemento i-1_esimo scambia
        {
            a[j+1]=a[j];
            j--;
        }
        a[j+1]=temp;
    }
}

```

```

C:\Windows\system32\cmd.exe

1o elemento: 10
2o elemento: 9
3o elemento: 8
4o elemento: 7
5o elemento: 1
6o elemento: 2
7o elemento: 3
8o elemento: 4
9o elemento: 5
10o elemento: 6

Array non ordinato:
1 9 8 7 1 2 3 4 5 6

Array ordinato:
1 1 2 3 4 5 6 7 8 9

Premere un tasto per continuare . . .

```

FINE

PIROMALLO ALESSIO