Esercizio sintesi (1)

Una compagnia di autobus che effettua servizio su lunghe distanze vuole realizzare un programma di controllo delle prenotazioni dei posti.

A tal scopo rappresenta ogni prenotazione tramite una struttura **`booking`** contenente nome del cliente (al massimo 1023 caratteri, senza spazi) e numero del posto prenotato (un intero).

Le prenotazioni effettuate vengono registrate tramite un array (di dimensione prefissata **`DIM`**) di strutture **`booking`**, di dimensione logica iniziale pari a 0.

Esercizio sintesi (1)

a) Si realizzi una funzione:

```
int assegna( booking list[],
             int * lengthList,
             char * name,
             int pref)
```

La funzione riceve in ingresso l'array di prenotazioni e la sua dimensione logica, e poi il nome del cliente ed il posto da lui indicato. La funzione deve controllare che il posto indicato non sia già stato assegnato, ed in caso contrario deve restituire il valore 1.

Esercizio sintesi (1)

Qualora invece il posto sia ancora libero, la funzione deve assegnare tale posto al cliente copiando i dati della prenotazione nell'ultima posizione libera nell'array, e deve provvedere ad aggiornare correttamente la dimensione logica dell'array. In questo secondo caso la funzione deve invece restituire come valore uno 0, indicante il successo nella prenotazione.

Al fine di copiare il nome del cliente, si utilizzi la funzine di libreria

```
char * strcpy(char * s, char * ct)
```

che copia ct in s (terminatore compreso).

Esercizio sintesi (1)

b) Si realizzi un programma main che chieda all'operatore il nome di un utente, e di seguito il posto prescelto. Il programma deve cercare di registrare la prenotazione tramite la funzione **`assegna`**; qualora l'operazione di prenotazione fallisca (perché il posto risulta essere già assegnato), il programma provveda a chiedere all'operatore un nuovo posto, finché non si riesca ad effettuare la prenotazione.

Esercizio sintesi (1)

Qualora l'operatore inserisca il nome "fine", il programma deve terminare; qualora invece venga inserita la stringa "stampa", il programma deve stampare a video le prenotazioni già effettuate.

A tal scopo si usi la funzione di libreria:

```
int strcmp(char * ct, char * cs)
```

che restituisce 0 se e solo se le due stringhe sono identiche (lessicograficamente).

Esercizio sintesi (1)
soluzione

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

#define MAX 1024
#define DIM 10


typedef struct {
     char name[MAX];
     int seat;
} booking;

...
```

Esercizio sintesi (1)
soluzione

```
int assegna(   booking list[], int * lengthList,
               char * name, int pref) {
    int i=0;
    int trovato = 0;

    while (( i < *lengthList) && !trovato) {
       if (list[i].seat == pref)
              trovato = 1;
       i++;
    }

...
```

Esercizio sintesi (1)
soluzione

```
...
    if (!trovato) {
       list[*lengthList].seat = pref;
       strcpy(list[*lengthList].name, name);
       (*lengthList)++;
       return 0;
    }
    else
       return 1;
}
```

Esercizio sintesi (1)
soluzione

```
int stampaBooking(booking list[], int lengthList) {
     int i=0;

     for (i=0; i<lengthList; i++)
        printf("%s: %d\n", list[i].name, list[i].seat);

     return 0;
}
```

Esercizio sintesi (1)
soluzione

```
int main() {
     booking list[DIM];
     int alengthList = 0;
     char nome[MAX];
     int seat = 0;

     printf("Inserire Nome Passeggero: ");
     scanf("%s", nome);
...
```

Esercizio sintesi (1)
soluzione

```
while (strcmp(nome, "fine")) {
   if (strcmp(nome, "stampa")) {
      printf("Posto preferito: ");
      scanf("%d%*c", &seat);

      while(assegna(list, &alengthList, nome, seat)) {
         printf("Spiacenti ma il posto scelto e' gia'
                  occupato.\n");
         printf("Per favore specificare un altro posto: ");
         scanf("%d%*c", &seat);
      }
   }
   else
          stampaBooking(list, alengthList);
```

Esercizio sintesi (1)
soluzione

```
...
        printf("Inserire Nome Passeggero: ");
        scanf("%s", nome);
    }

    return 0;
}
```

Esercizio sintesi (2)

Si vuole implementare un programma per il calcolo dell'inflazione su determinati prodotti commerciali.

A tal scopo ogni prodotto è rappresentato tramite una struttura **`item`**, definita da una stringa **`name`** con il nome del prodotto, e da due float **`old_price`** e **`new_price`** rappresentanti i prezzi.

Esercizio sintesi (2)

a) Si scriva una funzione **`lettura()`** che riceva come parametri di ingresso un vettore **`prezzi`** di strutture **`item`**, la dimensione fisica **`max`** del vettore **`prezzi`**, e un puntatore a intero **`num`** che rappresenta la dimensione logica del vettore. La funzione deve leggere da standard input il nome del prodotto ed i due prezzi, e deve copiare tale informazione nella prima posizione libera nel vettore **`prezzi`**.

Esercizio sintesi (2)

La funzione deve terminare se l'utente inserisce come nome del prodotto il termine "fine", oppure se viene raggiunta la dimensione fisica del vettore.

La dimensione logica del vettore **`prezzi`** così riempito deve essere restituita tramite il parametro **`num`** (passato appunto per riferimento). Al termine della lettura dei dati la funzione deve restituire il valore 0.

Esercizio sintesi (2)

b) Si scriva un programma **`main`** che, dopo aver definito un vettore di strutture **`item`** (di dimensione massima **`MAX_ITEM`**), invochi la funzione **`lettura()`** per riempire tale vettore.

Il programma stampi poi a video nome e tasso d'inflazione per ogni prodotto, utilizzando la formula:

$$infl_i = \left(\frac{new\_price_i}{old\_price_i} - 1 \right) * 100$$

Esercizio sintesi (2)
soluzione

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

#define DIM 21
#define MAX_ITEM 100

typedef struct {
     char name[DIM];
     float old_price;
     float new_price;
} item;
...
```

Esercizio sintesi (2)
soluzione

```
int lettura (item prezzi[], int max, int * num) {
     char name[DIM];
     *num = 0;

     printf("Inserire nome prodotto: ");   scanf("%s", name);

     while ((strcmp(name, "fine")) && (*num < max)) {
         strcpy(prezzi[*num].name, name);
         printf("Inserire old price: ");
         scanf("%f", &prezzi[*num].old_price);
         printf("Inserire new price: ");
         scanf("%f%*c", &prezzi[*num].new_price);

         (*num)++;

         printf("Inserire nome prodotto: ");
         scanf("%s", name);
     }   return 0;
}
```

Esercizio sintesi (2)
soluzione

```
int main() {
     item V[MAX_ITEM];
     int num, i, result;
     float infl;

     result = lettura(V, MAX_ITEM, &num);

     if (result!=0) {
        printf("Problemi durante la lettura...\n");
        exit(-1);
     }

     for (i=0; i < num; i++) {
        infl = (V[i].new_price/V[i].old_price -1)*100;
        printf("Inflazione del prodotto %s: %6.2f%%\n", V[i].name, infl);
     }
     return 0;
}
...
```

Esercizio sintesi (3)

Scrivere una funzione ricorsiva:

`int ric(int x)`

che calcoli, *ricorsivamente*, la somma di tutti i numeri compresi tra 0 ed **`x`**.

```
int ric(int x) {
    if (x == 0)
       return 0;
    else
       return x + ric(x-1);
}
```

Esercizio sintesi (4)

Scrivere una procedura ricorsiva:

```
void print(int list[], int length)
```

che stampi, *ricorsivamente*, tutti i numeri contenuti nell'array **`list`**.

```
void print (int list[], int length) {
    if (length == 0)
       return;
    else {
       printf("%d\n", * list);
        print(++list, --length);
    }
}
```

Esercizio sintesi (4bis)

Scrivere una procedura ricorsiva:

```
void print(int list[], int length)
```

che stampi, *ricorsivamente*, tutti i numeri contenuti nell'array **`list`**.

```
void print (int list[], int length) {
    if (length == 0)
       return;
    else {
       print(list, length-1);
       printf("%d\n", list[length-1]);
    }
}
```

Esercizio sintesi (5)

Scrivere una procedura ricorsiva:

```
void printchar(char stringa[])
```

che stampi, *ricorsivamente*, tutti i caratteri contenuti in **stringa**, un carattere per linea, assumendo che **stringa** sia *ben formata*.

```
void printchar (char stringa[]) {
    if (*stringa == '\0')
       return;
    else {
       printf("%c\n", * stringa);
       printchar(stringa+1);
    }
}
```

Esercizio sintesi (6)

Scrivere una procedura ricorsiva che, ricevuto in ingresso un array di interi, esegua la somma degli interi in posizione con indice dispari.

```
int sumOdd2(int list[], int length, int pos) {
    if (pos >=length)
       return 0;
    else
       return list[pos] + sumOdd2(list, length, pos+2);
}

int sumOdd(int list[], int length) {
    return sumOdd2(list, length, 1);
}
```

Esercizio sintesi (7)

Si scrivano le versioni ricorsiva ed iterativa (utilizzo di while) di una funzione:

```
double  f(double a, int n);
```

che calcoli il seguente valore:

$$\sum_{i=1}^{n}\left(a-\frac{i}{a}\right)$$

Esercizio sintesi (7)

```
double f(double a, int n)
{ if (n==1) return a - 1/a;
  else return a - n/a + f(a, n-1);
}

double f(double a, int n)
{ int i=1;
  double sum=0;
  while(i<=n)
      {sum = sum + a - i/a;
       i++;}
  return  sum;
}
```

Esercizio analisi (8)

```
void F(int v[], int pos) {
     int N=4;    N--;
     *(v+pos+1) = *(v+pos+1) + *(v+pos);   return; }

int main () {
     int N=4, v[5], i;
     {   ++N; }
     for (i=0; i<N; i++)
         v[i]=2*i+1;

     for (i=0; i<N-1; i++)
         if (v[i]) F(v, i);

     printf("Adesso N vale: %d\n", N);

     for (i=1; i++ < (i?N:0); )
         printf("%d\n", v[i-1]);

     return 0; }
```

Esercizio analisi (9)

```
#include <stdio.h>
#include <stdlib.h>
#define DIM 9

int Funz(char y[], int *N, int *M) {
     int i;
     (*N)++;
     for (i=(*N)-1; i<*N; i++)
        y[i] = y[*M];
     return *N; }
```

```
int main () {
     char s[] = "Paperone";
     char ss[DIM]; int i=3, N = 2;
     N = (++N)-2;
     N = Funz(s, &N, &i);
     printf("N vale adesso: %d\n",N);
     {
         for (i=0; i<DIM; ++i) {
                 *(ss+i) = s[i];
                 printf("%c", *(ss+i)); }
     }
     return 0;
}
```