RELAZIONE – LUCA MAZZUCCO S203372_020215
PROGRAMMAZIONE 12 PUNTI

ESERCIZIO 1

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

void eraseDuplicate(char*str)
{
   int i = 1, j=0, n=0, flag =0;

   char *str2 = malloc(sizeof(str));
   str2[0] = str[0];

   while(str[i]!='\0')
   {
      j = n;
      flag = 0;
      while(j>=0)
      {
         if(str[i]==str2[j])
            flag = 1;
         j--;
      }
      if(flag==0)
      {
         n++;
         str2[n] = str[i];
      }
      i++;
   }
   n++;
   str2[n] = '\0';
   printf("%s", str2);
   return;
}
```

La seguente funzione iterativa riceve come attributo la stringa str e utilizza la stringa str2 come risultato di pari lunghezza.

Un ciclo scorre tutti i caratteri della stringa str confrontandoli ad uno ad uno con i caratteri già immessi nella stringa str2.
Un flag segnala l'eventuale ritrovamento di un carattere uguale ad uno già caricato non permettendo così l'aggiunta di un nuovo carattere in str2.

Infine si aggiunge il carattere di fine stringa a str2 e si stampa il risultato.

A differenza della funzione scritta in esame è stato corretto soltanto il return della funzione che essendo void doveva essere vuoto.

ESERCIZIO 2

```
#include <stdio.h>
#include <stdlib.h>
#define N 10

int level =-1,  j;

void visitLevelByLevel(struct node *root, int l1,
int l2)
{
   int i;
   if(level == j)
   {
      printf("%d", root->key);
      return;
   }

   level++;

   for(i=0; i<N; i++)
   {
      visitLevelByLevel(root->children[i], l1,
l2);
   }
   level--;

   if(i == N && level == 0 && j<=l2)
   {
      j++;
      visitLevelByLevel(root, l1, l2);
   }

   return;
}
```

La seguente funzione ricorsiva è composta da:

- Una condizione di fine ricorsione, vera se il livello corrente ("level") è uguale al livello cercato ("j" inizializzato a l1 nel main) con eventuale stampa della chiave;

- Ciclo For per il richiamo della funzione;

- Una condizione vera se i parametri rispecchiano quelli della prima ricorsione e se j<=l2, la quale richiama nuovamente la funzione con il nuovo livello cercato (j++).

Rispetto alla versione della sede d'esame sono state apportate le seguenti modifiche:
1 – la riga level++ è stata portata fuori dal ciclo;
2 – le variabili "11" e "12" lette con distrazione sono state sostituite con "l1" e "l2" (una variabile chiamata con soli numeri non è permessa);
3 – per chiarezza sono state accorpate le ultime due condizioni e sostituito il parametro "tree->head" con "root", che nella prima chiamata di funzione è uguale alla radice dell'albero.

ESERCIZIO 3

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

void disp(int pos, char**set, char*sigla, int n,
FILE*f)
{
   int i;
   if (pos==n)
   {
      fprintf(f, "%s\n", sigla);
      return;
   }
   for(i=0; i<strlen(set[pos]); i++)
   {
      sigla[pos] = set[pos][i];
      disp(pos+1, set, sigla, n, f);
   }


   return;
}
```

La seguente funzione calcola tutte le disposizioni possibili con scelta variabile per ogni posizione.

E' composta da:
- Una condizione di terminazione, vera se sono già state inseriti tutti i caratteri;

-Un ciclo di chiamate alla funzione preceduto dall'assegnazione di un carattere alla stringa risultato "sigla".

Rispetto alla versione della sede d'esame sono state apportate le seguenti modifiche:
1 – Eliminata la variabile sol inutilizzata tra i parametri della funzione;
2 – Sostituito "sizeof" con "strlen" in quanto bisognava rapportare il valore alla dimensione del char;
3 – sostituito "=" con "==" nella prima condizione;