Lezione 3 Soluzione esercizi slides

Scrivere un'applicazione che data una stringa come argomento ne stampa a video la lunghezza, ad esempio: ./lengthof "Questa frase ha 28 caratteri" deve restituire a video il numero 28.

```c
/* lengthof.c */

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

int main(int argc, char **argv) {
  int code=0, len=0;
  char *p;
  if (argc!=2) {
    printf("Usage: %s <string>\n", argv[0]);
    code=2;
  } else {
    p=argv[1];
    while (*p !=0 ){
        p++;
        len++;
    }
    printf("%d\n", len);
    printf("%ld\n",strlen(argv[1]));
  }
  return code;
}
```

Scrivere un'applicazione che definisce una lista di argomenti validi e legge quelli passati alla chiamata verificandoli e memorizzando le opzioni corrette, restituendo un errore in caso di un'opzione non valida.

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define TRUE 1
#define FALSE 0
#define MAX_PARAMS 4
typedef char bool;

const char * valid[] = {"-h","-m","-n","--help"};

//Return TRUE is parameter is valid, FALSE otherwise
bool isValid(char * arg){
    for (int i = 0; i<4;i++){ // Cycle through all of the valid parameters
        //Compare the i-th element of the valid list with the parameter
```

```
        if(!strcmp(arg,valid[i])){
            return TRUE;
        }
    }
    return FALSE;
}

int main(int argc, char ** argv){
    char * paramList[MAX_PARAMS]; //Define list of stored parameters
    int ind = 0;

    for(int i = 1; i<argc; i++){ //Cycle through parameters
        if(isValid(argv[i])){ //Check if parameter is valid
            paramList[ind++]=argv[i]; //Save valid parameter
        }else{
            fprintf(stdout,"'%s' is a invalid parameter\n",argv[i]);
            exit(2);
        }
    }

    //Print list of recognised parameters
    for (int i = 0; i< ind; i++){
        printf("%s\n",paramList[i]);
    }
    return 0;
}
```

Realizzare funzioni per stringhe char *stringrev(*char str) (inverte ordine caratteri) e int stringpos(*char str, char chr) (cerca chr in str e restituisce la posizione)

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

#define MAX_LEN 50

//Invert the string. Return the pointer to the new string.
char * stringrev(char * str){
    char tmp[50]; //Declare temporary string
    short len = strlen(str); //Save string length
    int ind=len;
    for(int i=0; i<len;i++){ //Cycle through string chars
        tmp[i]=str[--ind]; // Save ind-th string element as the i-th
    }
    tmp[len]=0; //Save termination char
    str = tmp;
```

```
    return str;
}

//Find chr in str and return its position if found. Return -1 otherwise
int stringpos(char * str, char chr){
    short len = strlen(str); //Save string length
    for(int i = 0; i<len; i++){ //Cycle through string chars
        if(str[i]==chr) //Compare chars
            return i;
    }
    return -1;
}

int main(int argc, char ** argv){
    if (argc < 3){
        printf("Error. Correct syntax: %s <string> <char>\n",argv[0]);
        exit(2);
    }
    if (strlen(argv[1])>MAX_LEN){
        printf("String is too long. Max length: %d\n",MAX_LEN);
        exit(3);
    }
    if (strlen(argv[2])>2){
        printf("Char is not valid");
        exit(4);
    }
    printf("Char found in pos %d\nInverted: %s\n",stringpos(argv[1],argv[2][0]),string
rev(argv[1]));

    return 0;
}
```