#define **e** 2.71828182845  
  
#include **<math.h>**

**double** power(**double** base, **double** exponent){  
 **double** res=1;  
 **for**(**int** i=0 ; i<exponent ; i++){  
 res\*=base;  
 }  
 **return** res;  
}  
  
**double** absoluteValue(**double** x){  
 **if**(x>=0){  
 **return** x;  
 } **else**{  
 **return** -x;  
 }  
}  
  
**int** mod(**int** x, **int** y){  
 **return** x%y;  
}  
  
**double** Exp(**double** x){  
 **return** power(**e**,x);  
}  
  
#include **<string.h>**

**int** stringLength(**char** \*str){  
 **int** length=0;  
 **while**(\*str!=**'\0'**){  
 length++;  
 str++;  
 }  
 **return** length;  
}  
  
**void** stringCopy(**char** \*destination, **char** \*source){  
 **while**(\*source!=**'\0'**){  
 \*destination=\*source;  
 destination++;  
 source++;  
 }  
}  
  
**void** stringNCopy(**char** \*destination, **char** \*source, **int** n){  
 **for**(**int** i=0 ; i<n ; i++){  
 destination[i]=source[i];  
 }  
}  
  
**void** stringConcatenate(**char** \*destination, **char** \*source){  
 **int** k= stringLength(destination);  
 **for**(**int** i=0 ; i< stringLength(source) ; i++){  
 destination[k++]=source[i];  
 }  
 destination[k]=**'\0'**;  
}  
  
**void** stringNConcatenate(**char** \*destination, **char** \*source, **int** n){  
 **int** k= stringLength(destination);  
 **for**(**int** i=0 ; i<n ; i++){  
 destination[k++]=source[i];  
 }  
 destination[k]=**'\0'**;  
}  
  
**int** stringCompare(**char** \*str1, **char** \*str2){  
 **int** counter=0;  
 **for**(**int** i=0 ;; i++){  
 **if**(counter> stringLength(str1) || counter> stringLength(str2)){  
 **return** 0;  
 }  
 **if**(str1[i]!=str2[i] && str1[i]>str2[i]){  
 **return** 1;  
 }  
 **if**(str1[i]!=str2[i] && str1[i]<str2[i]){  
 **return** -1;  
 }  
 counter++;  
 }  
}  
  
**int** stringNCompare(**char** \*str1, **char** \*str2, **int** n){  
 **for**(**int** i=0 ; i<n ; i++){  
 **if**(str1[i]!=str2[i] && str1[i]>str2[i]){  
 **return** 1;  
 }  
 **if**(str1[i]!=str2[i] && str1[i]<str2[i]){  
 **return** -1;  
 }  
 }  
 **return** 0;  
}  
  
**char** \*Substring(**char** \*str, **char** \*subStr){  
 **int** lenStr= stringLength(str);  
 **int** lenSubStr= stringLength(subStr);  
 **int** i,j;  
 **for**(i=0 ; i<=lenStr-lenSubStr ; i++){  
 **if**(stringNCompare(subStr,str+i,lenSubStr)==0){  
 **break**;  
 }  
 }  
 **return** &str[i];  
}  
  
#include **<ctype.h>****int** isAlpha(**char** c){  
 **return** c<=**'Z'** && c>=**'A'** || c<=**'z'** && c>=**'a'**;  
}  
  
**int** isDigit(**char** c){  
 **return** c>=**'0'** && c<=**'9'**;  
}  
  
**int** isAlNum(**char** c){  
 **return** c<=**'Z'** && c>=**'A'** || c<=**'z'** && c>=**'a'** || c>=**'0'** && c<=**'9'**;  
}  
  
**int** isSpace(**char** c){  
 **return** c==**' '**;  
}  
  
**int** isLower(**char** c){  
 **return** c<=**'z'** && c>=**'a'**;  
}  
  
**int** isUpper(**char** c){  
 **return** c<=**'Z'** && c>=**'A'**;  
}  
  
**char** toLower(**char** c){  
 **if**(isUpper(c)){  
 **return** c+=32;  
 }  
 **return** c;  
}  
  
**char** toUpper(**char** c){  
 **if**(isLower(c)){  
 **return** c-=32;  
 }  
 **return** c;  
}  
  
**int** isVowel(**char** c){  
 **return** toLower(c)==**'a'** || toLower(c)==**'e'** || toLower(c)==**'i'** || toLower(c)==**'o'** || toLower(c)==**'u'**;  
}  
  
**int** isConsonant(**char** c){  
 **return** !isVowel(c);  
}