CS217 Object Oriented Programming
Assignment No. 1
February 9, 2019

Deadline: February 20, 2019 at 10:00 AM

Attention

- Make sure that you read and understand each and every instruction. If you have any questions or comments you are encouraged to discuss with your colleagues and instructors on Piazza.
- Submit a single '.zip' file for your assignment, and each problem solution must be provided in a separate CPP file. For instance, you must name the file containing solution of the first file as 'q1.cpp' and the second as 'q2.cpp' and so on.
- For all the problems, use " char * ", instead of string. The usage of string is strictly prohibited.
- **Please start early otherwise you will struggle with the assignment.**
- **You must follow the submission instructions to the letter, as failing to do so can get you a zero in the assignment.**
- **Note: Punishment for the plagiarism is to award a straight zero in this (or all) assignment. Both the parties involved in the plagiarism (with or without the knowledge of the plagiarism), will be considered equally responsible and be penalized.**

Q1: String manipulation functions

Write the implementation of the following functions:

```
1  int Strlen(char *s1)
2  /*Returns the length of the string in number of characters.*/
3  {
4  }
```

```
1  char *Strcpy( char *s1, const char *s2 )
2  /*Copies string s2 into array s1. The value of s1 is returned.*/
3  {
4  }
```

```
1  char *Strncpy( char *s1, const char *s2, size_t n )
2  /*Copies at most n characters of string s2 into array s1.
3  The value of s1 is returned.*/
4  {
5  }
```

```
1  char *StrCat( char *s1, const char *s2 )
2  /*Appends string s2 to array s1.
3  The first character of s2 overwrites
4  the terminating null character of s1.
5  The value of s1 is returned.*/
6  {
7  }
```

```c
char *StrnCat( char *s1, const char *s2, size_t n )
/*Appends at most n characters of string s2 to array s1.
The first character of s2 overwrites the terminating
null character of s1. The
value of s1 is returned.*/
{
}
```

```c
int StrCmp( const char *s1, const char *s2 )
/*Compares the string s1 with the string s2.
The function returns 0, less than 0 or greater
than 0 if s1 is equal to, less than or greater
than s2 , respectively.*/
{
}
```

```c
int StrnCmp( const char *s1, const char *s2, size_t n )
/*Compares up to n characters of the string
s1 with the string s2. The function returns 0,
less than 0 or greater than 0 if s1 is equal
to, less than or greater than s2, respectively.*/
{
}
```

```c
char **StrTok( char *s1, const char s2)
/*A call to StrTok breaks string s1 into
''tokens'' (logical pieces such as words
in a line of text) separated by character
contained in char s2*/
{
}
```

```c
int StrFind(char *s1, char *s2)
/*Searches the string s1 for the first occurrence
of the string s2 and returns its starting index,
if s2 not found returns -1.*/
{
}
```

```c
char * SubStr (char *, int pos, int len)
/*This function returns a newly constructed
string variable with its value initialized
to a copy of a substring of this variable.
The substring is the portion of the string
that starts at character position ''pos'' and
spans ''len'' characters (or until the end
of the string, whichever comes first).*/
{
}
```

Letters, numbers, punctuation

Character	Code	Character	Code	Character	Code	Character	Code	Character	Code
A	·−	J	·−−−	S	···	1	·−−−−	Period [.]	·−·−·−
B	−···	K	−·−	T	−	2	··−−−	Comma [,]	−−··−−
C	−·−·	L	·−··	U	··−	3	···−−	Question mark [?]	··−−··
D	−··	M	−−	V	···−	4	····−	Apostrophe [']	·−−−−·
E	·	N	−·	W	·−−	5	·····	Exclamation mark [!]	−·−·−−
F	··−·	O	−−−	X	−··−	6	−····	Slash [/], Fraction bar	−··−·
G	−−·	P	·−−·	Y	−·−−	7	−−···	Parenthesis open [(]	−·−−·
H	····	Q	−−·−	Z	−−··	8	−−−··	Parenthesis close [)]	−·−−·−
I	··	R	·−·	0	−−−−−	9	−−−−·	Ampersand [&], Wait	·−···

Figure 1: A mapping table between English Alphabets and Morse codes. Source: http://en.wikipedia.org/wiki/Morse_code

Q2: Morse Code

Morse code has been one of the most basic communication protocol and still used to convey SOS messages and other urgent communication. In Morse code each English alphabet is encoded by a sequence of '.' and '-', see Figure 1 for complete mapping between English alphabets and Morse codes. Letters are separated by spaces and words by "/". Your task is to design a program that can (i)convert any given string into a Morse code sequence *char* convertToMorseCode(char*)* and (ii) a Morse code sequence to a string. *char* convertToString(char*)* **You are not authorized to use *string* data type, however you can use *char****

Q3: Magic Squares

In this question your goal is to write code for solving magic square (for details read the attached file magic-square.pdf) using 2-D pointers. Your function should take the the dimension of magic square as input and then solve the magic square for given dimension. You are required to return a 2d pointer to an integer.

```
1 int** magicSquare (int size)
2 /*This function returns a 2d pointer that dynamically
3 allocates a matrix of size passed as arguments
4 and store magic square in allocated matrix.*/
5 {
6 }
```

Q4: Text Analysis

The availability of computers with string-manipulation capabilities has resulted in some rather interesting approaches to analyzing the writings of great authors. This exercise examines three methods for analyzing texts with a computer. **You have to use char * for following exercises.**

1. Write a function that receives a string consisting of several lines of text and returns an array indicating the number of occurrences of each letter of the alphabet in the text. For example, the phrase To be, or not to be: that is the question: contains one a, two bs, no cs, and so on.

```
void countLetters(char *string, int *&array, int & size)
/*Parameters:
Input:
char * : a multiline string
Output:
int *: an array containing counts of each letter,
to be allocated in function
int : array size */
{
}
```

2. Write a function that receives a string consisting of several lines of text and returns an array indicating the number of one-letter words, two-letter words, three-letter words, and so on, appearing in the text. For example, the phrase Whether this nobler in the mind to suffer contains 2, 3, 4, etc. length words.

```
void countWordsBasedOnLength(char *string, int *&array/*to be allocated */,
int & size/*updated array size*/)
/*Parameters:
Input:
char * : a multi-line string
Output:
int *: an array containing counts of each different length words,
to be allocated in function
int : array size */
{
}
```

3. Write a function that receives a string consisting of several lines of text and returns arrays indicating unique words and the number of occurrences of each unique word in the text along with their size.

```
void countingUniqueWords(char *string, char **&uwords/*list of unique words;*/,
int *&array/*to be allocated */, int & size/*updated array size*/)
/*Parameters:
Input:
char * : a multiline string
Output:
char **: an array of unique words
int *: their counts
int : number of unique words*/
{
}
```