# CS 103 Computer Programming

Assignment Number 1a

February 3, 2017

**Deadline:** Sunday 12th February, 2017 before 23h00

**Attention**

- Make sure that you read and understand each and every instruction. If you have any questions or comments you are encouraged to discuss your problems with your colleagues (and instructors) on Piazza.
- Plagiarism is strongly forbidden and will be very strongly punished. If we find that you have copied from someone else or someone else has copied from you (with or without your knowledge) both of you will be punished. You will be awarded straight zero in this assignment or all assignments.
- 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 first file as 'q1.cpp' and second as 'q2.cpp' and so on.
- For all the problems you are required to user "char *", usage of string is strictly prohibited.
- **Please start early otherwise you will struggle with it**.
- **Note: You have to follow the submission instructions to the letter. Failing to do so can get a zero in assignment.**

**Q1: 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. Your task is to design a program that can (i) convert any given string into a Morse code sequence and (ii) a morse code sequence to string. **You are not authorized to used** `string` **date type, however you can use** `char *`

**Q2: Hangman** In this question your goal is to write a program for playing Hangman game. You are given a dictionary of 4000 frequently occurring '(1-4000.txt)' English words and a function for file reading and extracting a list of strings. During game execution, your program will randomly choose any words from the list of read words and use it as a target guessing word. Your program must display $\frac{1}{3}$ randomly selected letters from the chosen word and ask the user to guess the remaining $\frac{2}{3}$ letters of the given word. The user can at most have twice the number of tries to the displayed blanks to guess the word correctly. On each wrong guess you will display some portion of the Hangman ASCII graphic given below. If the user is unable to guess the word in the given number of tries you can either finish the game or deduct his life count.

## 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 mappping table between English Alphabets and Morse codes. Source:`http://en.wikipedia.org/wiki/Morse_code`

```
// function for reading the text file
#include<fstream>
void fread(string fname, string words[], const int & nwords)
// returns the read string of words
// make sure that you give full path of file to be read
// nwords will contain total number of read words
// words will contain array of read words
{
        ifstream ifile(fname.c_str(),ios::in);
        if (!ifile)
        {
                cout<<" Couldn't read the file " << fname;
                exit(-1);
        }

        int count=0;
        while(ifile && count < nwords)
                ifile >> words[count++];
        ifile.close();
}
```

```
// ASCII art for Hangman as an array of strings
  char *hangman[19]={" ____________..________",
"|  .____________))______|",
"|  |  / /        ||",
"|  |/ /         ||",
"|  |  /          ||.-''.",
"|  |/           |/  _  \\",
"|  |            ||  `/,|",
"|  |            (\\\\\`_.'",
"|  |           .-`--'.",
"|  |          /Y  .  . Y\\",
"|  |         // |    | \\\\",
"|  |        //  |  .  |  \\\\",
"|  |      ')    |     |   (`",
"|  |            ||'||",
"|  |            ||  ||",
"|  |            ||  ||",
"|  |            ||  ||",
"|  |           /  |  |  \\",
"             |_`-' `-'  |  |"};
```

**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 program should ask the user for the dimension of magic square and then solve the magic square for given dimension. You must do the proper allocation and deallocation of the memory.

**Q4: String manipulation functions** Write the implementation of the following functions

```
int Strlen(char *s1)
/*returns the length of string in number of characters...*/
{
}
char *Strcpy( char *s1, const char *s2 )
/*Copies string s2 into array s1.
The value of s1 is returned.
char *strncpy( char *s1, const char *s2, size_t n )
Copies at most n characters of string s2 into array s1.
The value of s1 is returned.*/
{

}
```

```
char *StrCat( char *s1, const char *s2 )
/*Appends string s2 to array s1.
The first character of s2 overwrites
the terminating null character of s1.
The value of s1 is returned.*/
{

}

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.*/
{
}


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.*/
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.
*/
{
}
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*/
{}

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.
*/
{}

char * SubStr (char *, int pos, int len)
/*
This function returns a newly constructed
string object with its value initialized
to a copy of a substring of this object.

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).
*/
{}
```

**Q5: String Text Editor** [30 Marks]

Here your goal is to build a string text editor, which will allow its user to create simple text files and store them in primary memory (Remember you will be storing data in primary memory RAM not on hard disk, so you will not need file handling). Your text editor will have following main menu:

```
Press 1. To create a new file.
Press 2. To view an existing file via giving a file name.
Press 3. To edit an exisiting file via giving its name.
Press 4. To copy an exisiting  file to a new file.
Press 5. To delete an exisiting via its name.
Press 6. To view list of all files with the names.
Press 7. To Exit.
```

**File Creation** Whenever user will press 1. your program will ask for the new file name and number of text lines in the file. You will also tell the user that in any single line of file there can be maximum of 60 characters, if there are more than 60 characters in a single line the remaining characters should be moved to next line. While taking input from user your program should automatically display each line number on the terminal, you will be using this number during file editing for updating a particular line. For example, this is how your program output will editing a file.

```
1. I am writing a text file.
2. This is second line of my file and i am storing data
3. on line by line basis.
```

**File Editing** User should be able to edit a file by giving its name. When editing a file user can upgrade any single line by giving its number and your program will update the content of that line with the newly provided text by user. Furthermore, use can ask to replace a specific word, *e.g.* user can ask to replace all the instances of word "work" with "enjoy". Your program should find all the instances of work work and should replace it with "enjoy".

**File Copying** Your program will allow its user to create a copy of an existing file and he should be asked to provide a valid new file name.

**File Deleting** User should be allowed to delete a file via its name.

For completing this task you will need to make use of dynamic memory and multi-dimensional pointers. So please start early otherwise you will struggle with it.

**Q5: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 "b's," no "c's," 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 multiline 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*/
;;
```