GNG1106 Winter 2021 - Assignment 5

Due: March 12, 23:59

Instructions

This assignment is to be done INDIVIDUALLY. Use the following instructions to complete and submit this assignment.

- You will need to submit your assignment electronically to Brightspace. Prepare the following
  - An assignment report in a PDF file (this allows you to use your favorite editor to create the PDF file). For question 1, insert the programming model filled in as per the question instructions. You may fill in the programming model using drawing features of your editor or by hand on paper which is then scanned and inserted into your document. For Questions 2 and 3, insert in your assignment report the source code and the output from running the program for all test cases. You must submit your source code files for question 2 and 3.
- Place all your files (PDF file and C source code files) in a directory A5_xxxxxxxx where xxxxxxxx is your student number.
- Zip your PDF document and the C source files in a zip file with the name A5_xxxxxx.zip where xxxxxx is your student number.
- Submit the zip file before the assignment deadline via Brightspace. In Brightspace, navigate to the Assignment page and click on "Click to submit Assignment 5" to reach the assignment 5 submission folder. You can also select the Assignment tab to see the Assignment folder pages. The Brightspace video "Assignments" (found in the page https://documentation.brightspace.com/EN/le/assignments/learner/submit_assignments.htm) provides details to help you submit the zip file.
- The questions are provided in both PDF and Word files. You may use the Word file (please remove the questions) to enter your answers in the document. An rtf file is also provided so that you may edit the file with a word processor other than Word. Do save your file as a PDF file for submission.
- Do start the assignment soon and do **not** wait until the last minute. You will be more efficient with a number of smaller efforts over a few weeks before the deadline than one large effort just before the deadline.

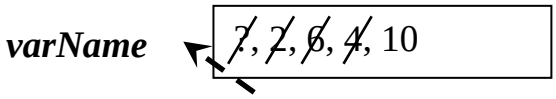
Marking Scheme (total 30 marks)

- Question 1: 5 marks
- Question 2: 10 marks
- Question 3: 15 marks

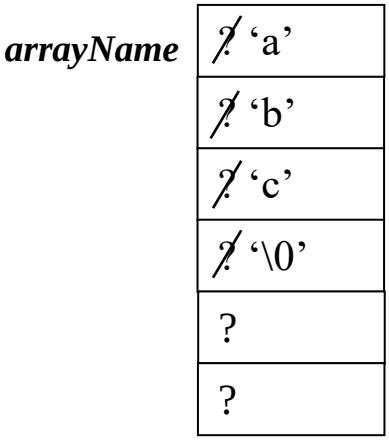
Question 1 (5 marks)

The following programming model contains in its code memory the indicated C program composed of two functions, `main` and `appendString` (this function is very similar to the standard `strcat` function). You will be showing how the working memory is used during the execution of this function. Show how the given C program affects the contents of the working memory:

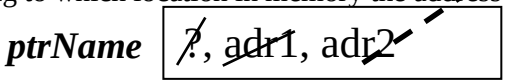
- Show how the arrays and variables occupy space in the working memory for each function.
- Show how the variables and arrays are initialized and updated by the program.
- Show the values are assigned the variable and elements in the array by the assignment instructions. Be sure to show all values that are assigned and replaced. Record successive assignments to variables/parameters as follows (the ? shows an unknown value):



- For arrays of type char, represent the array as shown below. Show the values in the array elements as literal value of type char.



- For the pointer variable, use adr1, adr2, adr3, etc. as address values assigned to the pointer variable and draw a dashed line showing to which location in memory the address is referring.



- Using arrows show how values are copied between the working memory allocated to the function **`main`** and the working memory allocated to the function `appendString`. It is **not** necessary to show how address values are copied to the parameters that are pointers (the dashed arrows will suffice).

Code Memory

```
#include <stdio.h>
#include <string.h>
#define SIZE_ARRAY_STR 15
// Prototypes
void appendString(char [], char []);
void main()
{
    char str1[SIZE_ARRAY_STR];
    char str2[SIZE_ARRAY_STR];
    // Initiliases the character arrays
    // using standard functions
    strcpy(str1, "The ");
    strcpy(str2, "string");
    // Appends the contents of the array
    // str2 to the end of the string
    // in the array str1
    appendString(str1, str2);
}
/*----------------------------
Description: The function appendsString appends the
             string referenced by s2 to the end
             of the string referenced by s1
             to the end of the string
-----------------------------*/
void appendString(char s1[], char s2[])
{
    int ix1, ix2;  // indexes of arrays
    // Finds the end of string 1
    // At the end of the loop,
    // s1[ix1] contains '\0'
    ix1 = 0;
    while(s1[ix1] != '\0') ix1 = ix1 + 1;
    // Now traverse string 2
    // to copy it into string 1
    for(ix2 = 0; s2[ix2] != '\0'; ix2 = ix2 + 1)
    {
        s1[ix1] = s2[ix2];
        ix1 = ix1 + 1;
    }
    s1[ix1] = '\0'; // to terminate the string
}
```

Working Memory

UCT

Question 2 (10 marks)

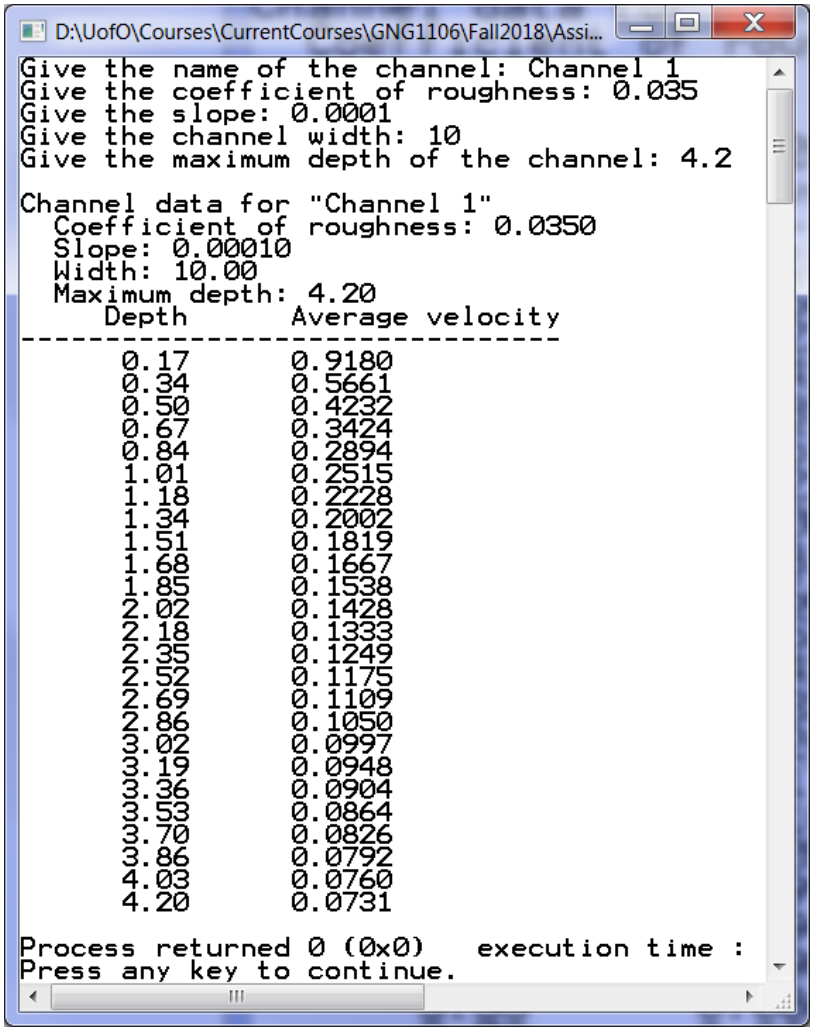
For this question, repeat question 2 from assignment 4, but this time, you will use a number of functions to separate well the various tasks of the program. Recall the question from assignment 4.

> The average velocity of water in a rectangular open channel can be calculated using the following Manning's equation.
>
> $$U = \frac{\sqrt{S}}{n}\left(\frac{B/H}{B+2H}\right)^{2/3},$$
>
> where U is the average velocity of the water (m/s),
> S is the channel slope
> n is the roughness coefficient ($s/m^{1/3}$)
> B is the width (m)
> H is the depth of the water (m)
>
> Develop a program that requests from the user characteristics of the channel and display in table form how the average velocity of the water varies with its depth in the channel.

Use the following new guidelines to develop your program.

- Definitions
  - Define a structure type CHANNEL that contains with members for the values for the open channel, that is, members `name` that holds a string for the name of the channel, `n` (roughness coefficient), `slope` (value of S), `width` (value of B), and `maxDepth` that holds the maximum water depth possible in the channel.
  - Use a symbolic constant to determine the number of lines to display in the table on the console. Use 25 lines in the table to display. Use other symbolic constants to eliminate magic numbers in your program.
- In the `main` function:
  - Declare a variable of type `CHANNEL` which will contain channel characteristics.
  - Declare either a two dimensional (2D) array or two one dimensional (1D) array which shall contain values of type `double`. In the case of the 2D array, the first row contains the water depth values and the second row contains average water velocity values. In the case of two 1D arrays, one array contains the water depth values and the other contains average water velocity values.
  - Get from the user values to initialize all the members of the structure variable of type `CHANNEL` by calling the function `getInput`.
  - Fill the 2D array (or the two 1D arrays) by calling the function `fillArray`.
  - Display on the console the desired output by calling the function `displayTable`.
  - Thus the main function contains only 3 calls to functions (in addition to declaring variables). Address of the structure variable and arrays are passed to the called functions.
- In the function `getInput`:
  - This function has only a single parameter, a pointer to a value of type CHANNEL.
  - Use the function `fgets` to initialise the member `name` (this function will place the complete line typed by the user including spaces as opposed to the `scanf` function that stops when it encounters a space). For the value of the other members of the structure variable (that are of type `double`), used the function `getPositiveValue` from the *CylinderVolumeLab5* project to ensure that a positive value is stored in each of these members. Be sure to use the `getPostiveValue` function from lab 5 and not the version from lab 4.
  - This function **DOES NOT** return any value.

- In the function `fillArray`:
  - This function has a pointer parameter to a value of type `CHANNEL`. It also has the necessary parameters to fill the 2D array (or the 1D arrays) declared in the `main` function.
  - The function shall fill in the array(s) with the values of depth and average velocity. It obtains the values of average velocity by calling the function `computeVelocity`. Note that Manning's equation cannot be applied to a depth with value 0. Thus the displayed table does not start at a depth of 0, but at the increment value used to increment the depth of the water.
- In the function `displayTable`:
  - This function has a pointer parameter to a value of type `CHANNEL`. It also has the necessary parameters to fill the 2D array (or the 1D arrays) declared in the `main` function.
  - The function displays on the console first the characteristics of the channel followed by a table of 25 lines that shows how the average speed of the water changes with its depth. The following shows an example of the desired output. Be sure to properly format the values.



- Function `computeVelocity`
  - This function has two parameters, one of type `double` which gives the depth of the water, and the second of type pointer to a value of type `CHANNEL` which contains the characteristics of the channel.
  - It computes the average velocity of the water using Manning's equation and returns this value.

- The following table gives three test cases to be used for testing your program.

Name	Channel 1
Roughness n (s/m^1/3)	0.035
Slope (m)	0.0001
Width(m)	10
Max. Depth (m)	4.2

Depth (m)	**Average Velocity (m/s)**
0.1680	0.917961
0.3360	0.566077
0.5040	0.423161
0.6720	0.342380
0.8400	0.289368
1.0080	0.251450
1.1760	0.222759
1.3440	0.200172
1.5120	0.181859
1.6800	0.166669
1.8480	0.153840
2.0160	0.142843
2.1840	0.133301
2.3520	0.124935
2.5200	0.117535
2.6880	0.110939
2.8560	0.105020
3.0240	0.099677
3.1920	0.094829
3.3600	0.090410
3.5280	0.086363
3.6960	0.082644
3.8640	0.079214
4.0320	0.076040
4.2000	0.073095

Name	Channel 2
Roughness n (s/m^1/3)	0.0013
Slope (m)	0.0032
Width (m)	2
Max. Depth (m)	11.5

Depth (m)	**Average Velocity (m/s)**
0.4600	56.740241
0.9200	29.778691
1.3800	19.693713
1.8400	14.450211
2.3000	11.266883
2.7600	9.146052
3.2200	7.641631
3.6800	6.524830
4.1400	5.666601
4.6000	4.988857
5.0600	4.441703
5.5200	3.991847
5.9800	3.616268
6.4400	3.298570
6.9000	3.026780
7.3600	2.791958
7.8200	2.587308
8.2800	2.407573
8.7400	2.248630
9.2000	2.107203
9.6600	1.980655
10.1200	1.866844
10.5800	1.764015
11.0400	1.670711
11.5000	1.585720

Name	Channel 3
Roughness n (s/m^1/3)	0.17
Slope (m)	0.041
Width (m)	40
Max. Depth (m)	1.5

Depth (m)	**Average Velocity (m/s)**
0.0600	7.756067
0.1200	4.876297
0.1800	3.713932
0.2400	3.059721
0.3000	2.631589
0.3600	2.325820
0.4200	2.094561
0.4800	1.912415
0.5400	1.764548
0.6000	1.641663
0.6600	1.537612
0.7200	1.448154
0.7800	1.370260
0.8400	1.301702
0.9000	1.240806
0.9600	1.186282
1.0200	1.137124
1.0800	1.092530
1.1400	1.051856
1.2000	1.014577
1.2600	0.980258
1.3200	0.948540
1.3800	0.919119
1.4400	0.891740
1.5000	0.866183

The answer to this question should provide in the assignment file file:

1) The source code to your program (also insert the source code into your assignment file).
2) The output for each test case in the above table into your assignment file.

Question 3 (15 marks)

Timber Regrowth

A challenge in timber management is to determine how many acres to leave uncut after harvesting in a forest so that the harvested area is reforested in a certain period of time. It is assumed that reforestation take place at a known constant rate, r, per year, depending on climate and soil conditions. A reforestation equation expresses this growth as a function of the amount of timber standing and the reforestation rate.

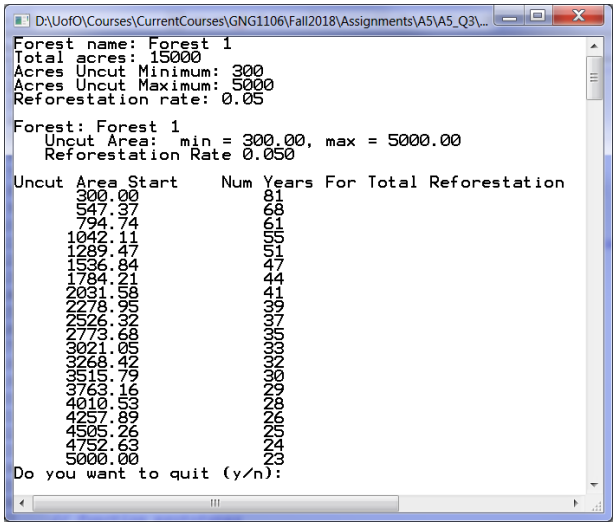
$$A_{i+1} = A_i + rA_i$$

where i represents the year, 0 after harvesting is completed, and 1, 2, 3 represents the number of years after harvesting (thus A_0 represents the acres uncut after harvesting is completed, and A_1, A_2, A_3, … represents the reforested acres after years 1, 2, 3, …).
r is the reforestation rate expressed as a fraction (for example 0.05 means that the forested acres are increases by 5% at the end of a year).

You are responsible for developing a software tool that supports analysis of reforestation. Given (from the user)

- the number of total acres,
- a minimum value for acres uncut,
- a maximum value for acres uncut,
- and reforestation rate,

show a table in the console that increments the values of uncut acres (from the given minimum to the maximum values) and for each of these values the total number of years it takes to totally restore the forest. The table shall contain 20 rows. The following shows a sample interaction with the user.



Complete the project using the following guidelines:

- Definitions
  - Define a structure type, `FOREST`, that contains 5 members, an array to store the name of the forest, and 4 members to store the data values obtained from the user (see the above list).
  - Use symbolic constants to eliminate magic numbers in your program.
- Function: `main`
  - Declares a structure variable of type FOREST and one or two arrays for storing table data.
  - Use a loop to repeat the process (get user input, fill in the table data and print a table) until the user wishes to quit.
  - This function shall call three functions to 1) get user input, 2) fill in the table data, and 3) display the table data.
- Function `getForestInput`

    - This function uses a pointer to fill in the structure variable declared in `main`.
    - Data input by the user must be checked so that
        - All real values must be greater than 0.0 (use the function `getPositiveValue` from lab 5).
        - The maximum uncut acres must be less than the total acres of the forest.
        - The minimum uncut acres must be less than the maximum uncut acres.
        - The reforestation rate must be between 0 and 1 (exclusive, i.e. cannot be 0 or 1).
- Function `calculateTableData`
    - The parameters of this function consist of a pointer to a structure variable of type `FOREST` and the necessary parameters to access and update the arrays declared in the function main. Do not use symbolic constants for the dimensions of the arrays, but parameters (this makes the function general to handle arrays of any dimension).
    - In addition to a loop to pass through the array (or arrays), use a nested loop to compute the number of years for total reforestation for a given value of uncut acres.
- Function displayTable
    - The parameters of this function consist of a pointer to a structure variable of type `FOREST` and the necessary parameters to access and update the arrays declared in the function main. Do not use symbolic constants for the dimensions of the arrays, but parameters (this makes the function general to handle arrays of any dimension).
    - This function first displays the forest information (see the sample output).
    - It then displays the contents of the array(s) in table form.
    - Please respect the output format such as the number of digits in the fraction part of the real numbers.

To answer this question, please provide:

1) The source code of your program (DO NOT insert the source code into your assignment document, PDF file).
2) Insert the output for the test cases shown in the table on the next page into your assignment document. Also show an output that shows how invalid input data is treated.

The following table gives the output for three tests cases (i.e. three ranges of time). The tables were computed using and Excel program (see the file *GNG1106A5TestCases.xlsx*).

Name	**Forest 1**		**Forest 2**		**Forest 3**	
Total Acres (km^2)	15000		22676		58732	
Uncut Acres Min (km^2)	300		1000		5000	
Uncut Acres Max (km^2)	5000		10000		20000	
Reforestation Rate	0.05		0.10		0.15	
Incr (km^2)	247.37		473.68		789.47	
	Uncut Acres	Years to reforest	Uncut Acres	Years to reforest	Uncut Acres	Years to reforest
1	300.00	81	1000.00	33	5000.00	18
2	547.37	68	1473.68	29	5789.47	17
3	794.74	61	1947.37	26	6578.95	16
4	1042.11	55	2421.05	24	7368.42	15
5	1289.47	51	2894.74	22	8157.89	15
6	1536.84	47	3368.42	21	8947.37	14
7	1784.21	44	3842.11	19	9736.84	13
8	2031.58	41	4315.79	18	10526.32	13
9	2278.95	39	4789.47	17	11315.79	12
10	2526.32	37	5263.16	16	12105.26	12
11	2773.68	35	5736.84	15	12894.74	11
12	3021.05	33	6210.53	14	13684.21	11
13	3268.42	32	6684.21	13	14473.68	11
14	3515.79	30	7157.89	13	15263.16	10
15	3763.16	29	7631.58	12	16052.63	10
16	4010.53	28	8105.26	11	16842.11	9
17	4257.89	26	8578.95	11	17631.58	9
18	4505.26	25	9052.63	10	18421.05	9
19	4752.63	24	9526.32	10	19210.53	8
20	5000.00	23	10000.00	9	20000.00	8