**CptS 121 – Program Design and Development**

**Spring 2021**

***Exam 2***

Friday, April 2, 2021

**Your Name: \_\_\_\_\_\_Duy Pham\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**ID Number: \_\_\_\_\_\_\_011742984\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**TA’s Name: \_\_\_\_\_\_\_ Aayush Bajoria \_\_\_\_\_\_\_\_\_\_\_**

READ THE FOLLOWING INSTRUCTIONS:

This exam is take-home. It should take ~ 1 hour to complete. You must work **individually** on this exam. You may use your book and notes if necessary. Please either handwrite or type your answers into this document. Late exam solutions will **not** be accepted! **Please show all work!!! Also, finding a solution online for any given problem, without properly citing it, could lead to zero credit for the problem.**

You must submit the exam through Blackboard by Monday, April 5, midnight PST. Be sure to answer each question precisely. Do not provide superfluous details in your answers. *NOTE: you do not need to comment your code solutions.*

|  |  |  |
| --- | --- | --- |
| **Part** | **Points Possible** | **Points Earned** |
| I | 50 |  |
| II | 50 |  |
| **Total** | **100** |  |

***Part I: Conceptual Questions (50 pts) – Short answer, Fill-in-the-blank, and True/False***

1. (4 pts) What is a *structure in C*? Explain.

A structure is a user defined data type in C. It is a way to combine multiple variables

into a single type. It is most often used in applications that work with databases; for

example, a database of students in a course, or employee records,...

1. (4 pts) What is an *array*? Explain.

An array is a collection of data items, all of the same type, accessed using a common

name, but a different index. The items are contiguously allocated in memory.

We can use an array to store related information like student ID numbers in a class,

and many more...

1. (4 pts) What is a C string? Explain.

A string is a sequence of characters terminated by the null character (‘\0’). The

characters are letters, digits, or special characters. A string may always be

represented by a character array, but a character array is not always a string.

For example: char name[20] = {‘A’, ‘n’, ‘n’, ‘a’, ‘ ‘, ‘K’, ‘i’, ‘m’, ‘\0’};

The name has 8 characters and a null character. 11 remaining characters after null character are ignored.

name[ ] = “Anna Kim”.

1. (2 pts) A(n) \_\_\_\_\_\_struct\_\_\_\_\_\_\_ is a collection of related *fields* or variables under one name. These may be used to describe real world *objects*.
2. (2 pts) A(n) \_\_\_ input validation\_\_\_ loop is a *pattern* in which the number of loop repetitions is generally not known before loop execution. The condition relies on a *special* value to terminate the loop.
3. (4 pts) The act of joining two strings is called \_\_\_\_string cat\_\_\_. The string library function that performs the joining is called \_\_\_\_strcat( )\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
4. (2 pts) A(n) \_\_\_\_\_\_data\_\_\_ *structure* is a way of storing and organizing information. An array is an example of one of these.
5. (2 pts) A(n) \_\_\_\_\_\_\_\_pointer\_\_\_\_\_\_ is a variable the stores the address of another variable.
6. (2 pts) What is the direct value of a pointer? \_\_The address of the variable pointed to\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
7. (16 pts) Answer the following questions (a.-f.) based on the array declaration below. **NOTE: operations for questions b-f are cumulative:**

char color[] = "red", building[20] = "house";

* 1. (6 pts - 0.5 pts/rectangle) Fill in each empty rectangle below with the appropriate contents for the array named color. You may choose an appropriate *starting* address.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 |
| Contents | ‘r’ | ‘e’ | ‘d’ | NULL |
| Address | 100 | 101 | 102 | 103 |

* 1. (2 pts) The contents of building[4] after the call strcpy(building, color)

is \_\_\_\_\_ ‘e’ .

* 1. (2 pts) The value of strlen(building) is \_\_\_3\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
  2. (2 pts) The value of strcmp(color, building) is: negative, positive, or zero? Zero
  3. (2 pts) The contents of color after the call strncpy(color, "blue", 2)

is \_\_\_\_\_\_\_\_\_\_\_\_\_\_”bld”\_ \_\_\_\_\_\_ .

* 1. (2 pts) The contents of building[3] after the call

strcat(building, &color[1])is:\_\_\_\_\_\_\_’l’\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

1. (2 pts) True / False: An *input validation* loop is a *pattern* which is used to repeat statements as long as a user entered value is not within a valid range. A *do-while* loop is best suited for this pattern. (True)
2. (2 pts) True / False: array[index] is equivalent to (array + index). (False)
3. (2 pts) True / False: A *string* is always represented by a character *array*, and a character array must *always* be manipulated as a string. (False)
4. (2 pts) True / False: In *C*, when an array is passed to a function, a copy of all of the elements in the array is made and passed into the function. (False)

***Part II: Programming Questions (50 pts) – Write C code***

1. (8 pts) Write a function called calculate\_grade\_point() that accepts a single *letter grade* (a character) and returns the associated *grade point* (a double). Please do *not* use any *output* parameters (i.e. pointers). You are required to solve this problem using a switch() statement. Be sure to include a default case as part of the switch() statement. Precondition: the letter grade must be uppercase. The table below shows the conversion from grade points to letter grades:

|  |  |
| --- | --- |
| **Letter Grade** | **Grade Points** |
| ‘A’ | 4.0 |
| ‘B’ | 3.0 |
| ‘C’ | 2.0 |
| ‘D’ | 1.0 |
| ‘F’ | 0.0 |

double calculate\_grade\_point (char letter\_grade)

{

double grade\_point = 0.0;

switch (letter\_grade)

{

case ‘A’ : grade\_point = 4.0;

printf(“Your grade point is %.1lf!\n” , grade\_point);

break;

case ‘B’ : grade\_point = 3.0;

printf(“Your grade point is %.1lf!\n” , grade\_point);

break;

case ‘C’ : grade\_point = 2.0;

printf(“Your grade point is %.1lf!\n” , grade\_point);

break;

case ‘D’ : grade\_point = 1.0;

printf(“Your grade point is %.1lf!\n” , grade\_point);

break;

case ‘F’ : grade\_point = 0.0;

printf(“Your grade point is %.1lf!\n” , grade\_point);

break;

default: /\* Anything but 'A', 'B', 'C', 'D', 'F' \*/

printf("Unrecognized Letter Grade!\n");

break;

}

return grade\_point;

}

1. (10 pts) Write a function called count\_vaccinations() that accepts an array of characters (that is NOT a string) and the corresponding number of residents for a county. The array contains a sequence of yes’ (represented by ‘y’ or ‘Y’) and no’s (represented by ‘n’ or ‘N’) that indicate which residents have received a vaccine shot. A ‘y’ or ’Y’ means the person received a shot, a ‘n’ or ‘N’ means the person did not receive a shot. This function returns the number of people who have been vaccinated. If you need a loop for this problem, then you must use a while loop to solve this problem.

int count\_vaccinations (char responses[ ], int num\_residents)

{

int num\_vaccinations = 0 ;

int i = 0 ; //index

while ( i < num\_residents)

{

if ( responses[i] == ‘y’ || responses[i] == ‘Y’ )

{

++ num\_vaccinations;

}

++i;

}

return num\_vaccinations;

}

1. (10 pts) Write a function called is\_duplicate() that accepts an integer array and the number of items in the array as parameters. The function returns 1 if a duplicate number exists in the array; 0 otherwise. A duplicate number is considered a number that occurs more than once in the array.

int is\_duplicate(int arr[ ], int size)

{

int i, j; //index

int boolean = 0; //1 is true, 0 is false

for (i = 0; i < size - 1; ++i)

{

for (j = i + 1; j < size; ++j)

{

if (arr[ j ] == arr[ i ])

{

boolean = 1; break;

}

}

if (boolean == 1) { break; }

}

return boolean;

}

1. (10 pts) Write a function called int\_to\_string() that accepts a 0 or positive integer, called num, and a pointer to a character array, called str\_ptr, as parameters. The function does not return a value. The function will convert the integer to a string representation. The integer 78 would be converted to string “78”. The resultant string is stored in the memory referenced by str\_ptr. You may use any library functions that you see fit. If you need a loop for this problem, then you must use a for loop Hint: you need to isolate each digit in the integer.

#include <math.h> //for using pow( )

void int\_to\_string(unsigned int num, char\* str\_ptr)

{

/\*First, we will determine the number of digits of the integer number.

Then it would equal the string length

(still not include the null character at the end yet)\*/

int num\_digits = 0, str\_len = 0;

int i; //index

if (num == 0) { num\_digits = 1; }

else

{

for (i = 0; pow(10, i) <= num; i++)

{

++num\_digits;

}

}

str\_len = num\_digits;

//Second, we convert each digit of the integer to each corresponding item in the string (from the final position to the first one)

\*(str\_ptr + str\_len) = NULL; //the null character at the end

for (i = str\_len - 1; i >= 0; --i)

{

\*(str\_ptr + i) = (num % 10) + 48; //Because char "k" corresponds to int k + 48 in the ASCII table (k is a letter from 0 to 9)

num /= 10;

}

}

1. (12 pts) Recall the binary search algorithm.

*Input:* a list of n *sorted* values and a target value

*Output*: True if target value exists in list and location of target value, false otherwise

*Method*:

Set left to index 0 and right to index n - 1

Set found to false

Set targetindex to -1

While found is false and left is less than or equal to right

Set mid to midpoint between left and right

If target = item at mid then set found to true and set targetindex to mid

If target < item then set right to mid – 1

If target > item then set to left to mid + 1

Return the targetindex

Write a C function for binary search. Your version of the binary search function must accept a 2D array of characters as a parameter. Where each row represents a string. The array consists of 15 rows and 100 columns. The function must also accept a target string as a parameter. The function must return the row index of where the target is found in the array. If the target is not found, then -1 is returned. You may use functions from <string.h>.

#include<string.h>

#define num\_row 15

#define num\_col 100

int binary\_search(char arr[][100], char target[])

{

//Firstly, we need to sort the strings by using strcmp()

int i, j; //index

char temp[10];

for (i = 0; i < num\_row-1; ++i)

{

for (j = i+1; j < num\_row; ++j)

{

if (strcmp(arr[i], arr[j]) == 1)

{

//swap the two strings by strcpy()

strcpy(temp, arr[i]);

strcpy(arr[i], arr[j]);

strcpy(arr[j], temp);

}

}

}

//Then, begin to search the target string

int found, left, right, mid;

found = 0; /\* false \*/

left = 0;

right = num\_row;

while (found == 0 && left <= right)

{

mid = (left + right) / 2;

if (strcmp(target, arr[mid]) == 0)

{

found = 1;

}

else if (strcmp(target, arr[mid]) == -1)

{

right = mid - 1;

}

else // strcmp(target, arr[mid]) == 1

{

left = mid + 1;

}

}

if (found == 1)

return mid; //he row index of where the target is found in the array

else

return -1;

}