ASSIGNMENT 2

This assignment is about writing C program to perform various tasks. Each task is performed by a function. The functions are called by the main. The functions and the main are in one source file named program.c. The program must compile under the gcc compiler at the command line. The program must run at the command line. The program must be well documented with comments explaining all steps.

1. Write a function that computes the value of the mathematical constant π. The value of π can be approximately computed using the infinite series $\pi = 4(1 - 1/3 + 1/5 - 1/7 + 1/9 - 1/11 + 1/13 - 1/15 + \ldots.)$ The function prototype is `double compute_pi(int n)`. The function computes the value of π using the first `n` terms of the infinite series and returns the value.

2. Write a function that computes the square root of a number. The square root of a number `x` can be approximately computed as follows. First guess that the square root of `x` is 1. Then repeatedly get the next guess from the last guess using the rule `next = 0.5(last + x / last)` where `last` is the last guess and `next` is the next guess. Repeat ten times using a loop and the tenth guess will be approximately the square root. The function prototype is `double compute_sqrt(double x)`. The function computes the square root of `x` and returns the square root.

3. Write a function that decides whether a number is a prime or not. A prime number is a number that is divisible only by 1 and itself. The function prototype is `int is_prime(int n)`. The function returns true if `n` is prime and returns false otherwise. Write another function that displays all prime numbers less than or equal to a number. The function prototype is `void display_primes(int n)`. The function displays all prime numbers less than or equal to `n`. Note that the `display_primes` function calls the helper function `is_prime`.

4. Write a function that reads student names and their scores from a user and displays the following: the average score, the minimum score, the maximum score, and the students who received the minimum and maximum scores. The function repeatedly prompts the user to enter the names and the scores of students. The user enters the name and the score separated by blank in one line. The name is a single word and the score is a positive integer. The user enters `q` in lower or upper case to quit. The user does not enter score after `q`. The outputs are displayed with appropriate messages. The function prototype is `void process_scores( )`. Arrays should not be used.

5. Write a function that determines the tax amount according to the following tax rules. The tax rate depends on the income, the marital status, and the state residency. For in state residents, the following rates apply. If single and income is less than 30000 then tax rate is 20%. If single and income is greater or equal to 30000 then tax rate is 25%. If married and income is less than 50000 then tax rate is 10%. If married and income is greater or equal to 50000 then tax rate is 15%. For out of state residents, the similar rules apply except the tax rate is 3% less than the tax rate of the corresponding in state residents. The function prototype is `double compute_tax(int income, char *status, char state)`. The income is positive integer. The status is string

`single` or `married` in lower or upper case letters. The state is character `i` or `o` in lower or upper case letters. The function computes and returns the tax amount. If any input is invalid then the function returns -1.

6. Write a function that solves a quadratic equation. The function prototype is `int quadratic(double a, double b, double c, double *solution1, double *solution2)`. The input parameters `a`, `b`, and `c` represent the coefficients of the quadratic equation $ax^2 + bx + c = 0$. The output parameters `solution1` and `solution2` represent the two solutions. First the function checks whether the equation actually has solutions. The equation has solutions only if $b^2 - 4ac \geq 0$. If there are solutions then the function computes the solutions using the formula $(-b \pm \sqrt{(b^2 - 4ac)})/2a$ and puts them into the output parameters. If there are no solutions then the function assigns `0` to the output parameters. The function returns true or false depending on whether the equation has solutions or not.

7. Write a recursive function that computes the factorial of a number. The factorial of a number `n` is the product `1.2.3.4...n` of the first `n` positive integers. The function prototype is `int factorial(int n)`. The function computes the factorial of `n` and returns it. The function must be recursive and loops should not be used.

8. Write a function that counts the characters and lines in a file. A character can be any character including letter, digit, symbol, blank, tab, newline, etc. A line is a sequence of characters terminated by newline character or end of file character. The function prototype is `void file_count(char *file, int *characters, int *lines)`. The function takes the file name as the input parameter and puts the counts of characters and lines into the output parameters.

9. Write a function that reads an input file and creates a sorted output file. The function prototype is `void file_sort(char *infile, char *outfile)`. The function takes the file names as parameters. The input file contains student information and has the following format. The first line contains the number of students in the file. Each subsequent line contains information about one student. Each line contains three fields namely student id, grade, and gpa in that order separated by blanks. The student id is a positive integer. The grade is a character. The gpa is a double type value. The function sorts the student information in the ascending order of student id. The ordered student information is written to the output file. The output file has the same format as the input file. The function dynamically allocates three arrays and stores the student information into the arrays. The three arrays are simultaneously sorted. The sorted array information is written to the output file. The dynamic arrays are freed when the function completes its work. Structures should not be used.

10. Write a function that processes a file containing student information. The function prototype is `void file_student(char *infile)`. The function takes the input file name as parameter. The input file has the following format. The first line contains the number of students in the file. Each subsequent line contains information about one student. Each line contains three fields namely name, age, and gpa in that order separated by blanks. The name is a single word. The age is an integer. The gpa is a double type value. A structure type representing a student is defined. An array of student structures is

dynamically allocated. The length of the array is the same as the number of students in the file. The student information is stored into the array of structures. Then the following tasks are performed. Print the average gpa of all students. Print the names of those students whose gpa is at least 2. Print all the student information in the ascending order of names. The dynamic array is freed when the function completes its work. No other array can be used.

11. Write a main program that allows a user to use the functions in questions 1-10. The program contains an interactive menu. The menu has 10 options. Each option is on separate line with an option number and an option description as follows. `1-computing pi, 2-computing square root, 3-displaying primes, 4-processing grades, 5-computing tax, 6-solving quadratic, 7-computing factorial, 8-counting file, 9-sorting file, 10-student file, 11-quit`. The user enters an option number. Each option calls its function. If the function needs inputs then the menu program asks the user for inputs. If the function gives outputs then the menu program displays the outputs with appropriate messages. The menu program repeatedly displays the menu and asks for an option until the user quits.

REQUIREMENTS FOR PROGRAMMING ASSIGNMENTS

1) Doing the assignment

Programming assignment must be done in the specified language

Program must compile at the command line unless specified otherwise

Program must run at the command line unless specified otherwise

Submit only source files

2) Submitting the assignment

Copy the source files into a usb disk

Print all the source code

Place the disk and the printout in a 9" × 12"envelope

Place any other required material in the envelope

Write your name on the printout and the envelope

Submit the envelope to the instructor on the due date at the beginning of class

3) Other matters about the assignment

Program must be written in good style

Program must be well documented with comments explaining all steps

Program that does not compile will receive zero grade

Print the program from a programming editor

Printout should not have lines wrapped around