#include <stdio.h> MAIN.C FILE

#include <assert.h>

#include <stdlib.h>

#include <string.h>

#include "practice\_problems.h"

int main(){

/\*

// 1.

// printf("A = %d, Z = %d, a = %d, z = %d", 'A', 'Z', 'a', 'z');

assert(upper\_to\_lower('A') == 'a'); // Assertion passes

// assert(upper\_to\_lower('B') == 'a'); // Assertion fails

assert(upper\_to\_lower('1') == '!'); // Assertion Returns !

assert(upper\_to\_lower('Z') == 'z');

assert(upper\_to\_lower('H') == 'h');

\*/

/\*

// 2.

printf("Returned Value = %f, Calculated Value = %f\n", area\_of\_circle(5.2), 3.14\*5.2\*5.2);

// Assertion will work fine only with int and char types

assert(area\_of\_circle(5.2) == 3.14\*5.2\*5.2); // Precision causes problem

\*/

/\*

// 3.

printf("%d", char\_between\_char('A', 'E'));

assert(char\_between\_char('Z', 'A') == 24);

assert(char\_between\_char('h', 'o') == 6);

\*/

/\*

// 4.

printf("Returned Value = %f, Calculated Value = %f\n", celsius\_to\_fahrenheit(38.0), (38.0 \* 1.8 + 32));

assert(celsius\_to\_fahrenheit(38.0) == (38.0 \* 1.8 + 32)); // Precision causes problem

\*/

/\*

// 5.

assert(odd\_or\_even(37) == 1); // Number is odd

assert(odd\_or\_even(24) == 0); // Number is even

printf("Returned Value = %d, Calculated Value = %d\n", odd\_or\_even(-5), 1);

assert(odd\_or\_even(-5) == 1); // Returns -1, needs to be fixed

assert(odd\_or\_even(24) == 1); // Number is even assertion fails

\*/

/\*

// 6.

assert(check\_leap\_year(2020) == 1); // Is a leap year

assert(check\_leap\_year(2000) == 1); // Is a leap year

assert(check\_leap\_year(1900) == 0); // Not a leap year

assert(check\_leap\_year(2022) == 0); // Not a leap year

\*/

/\*

// 7.

assert(pow\_of\_2(5) == 32);

assert(pow\_of\_2(10) == 1024);

assert(pow\_of\_2(1) == 2);

assert(pow\_of\_2(0) == 1);

\*/

/\*

// 8.

assert(char\_or\_int('5') == 1); // Integer

assert(char\_or\_int('\*') == -1); // Integer

assert(char\_or\_int('M') == 0); // Integer

assert(char\_or\_int('g') == 0); // Integer

\*/

/\*

// 9.

unsigned int sum = 0;

unsigned int x = rand();

unsigned int y = rand();

sum = x + y;

assert(add\_rand(x, y) == sum);

\*/

/\*

// 10.

assert(sum\_of\_digits(12345) == 15);

assert(sum\_of\_digits(752904) == 27);

assert(sum\_of\_digits(3) == 3);

\*/

/\*

// 11.

assert(reverse\_of\_num(5) == 5);

assert(reverse\_of\_num(57903) == 30975);

assert(reverse\_of\_num(500) == 005);

\*/

/\*

// 12.

assert(count\_digit(123212, 2) == 3);

assert(count\_digit(1, 1) == 1);

assert(count\_digit(3000007, 0) == 5);

\*/

/\*

// 13.

assert(check\_palindrome(12321) == 1);

assert(check\_palindrome(56788765) == 1);

assert(check\_palindrome(12322) == 0);

\*/

/\*

// 14.

assert(is\_prime(7) == 1);

assert(is\_prime(27) == 0);

assert(is\_prime(2) == 1);

unsigned int num;

unsigned int i, count=1;

printf("Enter the Number of Prime Numbers Required: ");

scanf("%d", &num);

for(i=2; count <= num; i++){

if(is\_prime(i)){

printf("%d\t", i);

count++;

}

}

\*/

/\*

// 15.

unsigned int num;

printf("Enter the Number of Elements Required: ");

scanf("%d", &num);

sum\_of\_series(num);

\*/

/\*

// 16.

assert(check\_if\_armstrong(153) == 1);

assert(check\_if\_armstrong(25) == 0);

\*/

/\*

// 17.

assert(check\_if\_amicable(220, 284) == 1);

assert(check\_if\_amicable(284, 220) == 1);

assert(check\_if\_amicable(10, 15) == 0);

\*/

/\*

// 18.

assert(menu\_driven\_calculator(7, 3, '+') == 10);

assert(menu\_driven\_calculator(7, 3, '-') == 4);

assert(menu\_driven\_calculator(7, 3, '\*') == 21);

assert(menu\_driven\_calculator(17, 3, 't') == -1);

\*/

/\*

// 19.

main\_menu();

\*/

/\*

// 20.

float units;

printf("Enter Electricity Consumed in Units: ");

scanf("%f", &units);

printf("Electricity Bill in Rupees: %.2f", electricity\_bill(units));

\*/

/\*

// 21.

// Binary to Decimal Conversion

assert(bin\_2\_dec(1011) == 11);

assert(bin\_2\_dec(10110010) == 178);

assert(bin\_2\_dec(1111) == 15);

assert(bin\_2\_dec(00) == 0);

assert(bin\_2\_dec(1) == 1);

// Decimal to Binary Conversion

assert(dec\_2\_bin(5) == 101);

assert(dec\_2\_bin(0) == 0);

assert(dec\_2\_bin(1) == 1);

assert(dec\_2\_bin(11) == 1011);

assert(dec\_2\_bin(47) == 101111);

\*/

/\*

// 22.

#define SIZE 100

int num;

int arr[SIZE] = {0, 0, 1};

printf("Enter the Number of Elements Required: ");

scanf("%d", &num);

generate\_series(arr, num);

for(int i=0; i<num; i++){

printf("%d ", arr[i]);

}

\*/

/\*

// 23.

unsigned int num;

printf("Enter the Number of Elements Required: ");

scanf("%d", &num);

print\_pattern\_1(num);

\*/

/\*

// 24.

unsigned int num;

printf("Enter the Number of Elements Required: ");

scanf("%d", &num);

print\_pattern\_2(num);

\*/

/\*

// 25.

generate\_series\_2();

\*/

/\*

// 26.

assert(power(2, 5) == 32);

assert(power(5, 3) == 125);

assert(power(2, 12) == 4096);

assert(power(7, 4) == 2401);

\*/

/\*

// 27.

assert(factorial(5) == 120);

assert(factorial(8) == 40320);

\*/

/\*

// 28.

printf("%f", series\_evaluation(3, 3));

\*/

/\*

// 29.

assert(CONCATENATE(56, 2) == 562);

assert(CONCATENATE(45, 763) == 45763);

\*/

/\*

// 30.

assert(SQUARE(7) == 49);

assert(SQUARE(9) == 81);

\*/

/\*

// 31.

main\_menu\_math\_operations();

\*/

/\*

// 32.

#define SIZE 5

int arr[SIZE] = {34, 13, 65, 8, 9};

print\_int\_array(arr, SIZE);

ascending\_sort(arr, SIZE);

print\_int\_array(arr, SIZE);

descending\_sort(arr, SIZE);

print\_int\_array(arr, SIZE);

\*/

/\*

// 33.

int x = 10, y = 25;

swap\_by\_ref(&x, &y);

assert(x == 25 && y == 10);

swap\_by\_ref(&x, &y);

assert(x == 10 && y == 25);

\*/

/\*

// 34.

#define SIZE 10

int arr[SIZE] = {34, 13, 65, 8, 9, 26, 47, 2, 9, 17};

int min, max;

min\_max\_array(arr, SIZE, &min, &max);

assert(min == 2);

assert(max == 65);

\*/

/\*

// 35.

#define SIZE\_A 10

#define SIZE\_B 10

#define SIZE\_C 20

int size\_c;

int arr\_a[SIZE\_A] = {34, 13, 65, 8, 9, 26, 47, 2, 9, 17};

int arr\_b[SIZE\_B] = {43, 31, 56, 8, 9, 62, 74, 2, 9, 71};

int arr\_c[SIZE\_C] = {0};

printf("Array A:\t\t");

print\_int\_array(arr\_a, SIZE\_A);

printf("Array B:\t\t");

print\_int\_array(arr\_b, SIZE\_B);

size\_c = set\_union(arr\_a, SIZE\_A, arr\_b, SIZE\_B, arr\_c, SIZE\_C);

assert(size\_c == 15);

printf("A Union B:\t\t");

print\_int\_array(arr\_c, size\_c);

size\_c = set\_intersection(arr\_a, SIZE\_A, arr\_b, SIZE\_B, arr\_c, SIZE\_C);

assert(size\_c == 3);

printf("A Intersection B:\t");

print\_int\_array(arr\_c, size\_c);

size\_c = set\_difference(arr\_a, SIZE\_A, arr\_b, SIZE\_B, arr\_c, SIZE\_C);

assert(size\_c == 6);

printf("A Minus B:\t\t");

print\_int\_array(arr\_c, size\_c);

size\_c = set\_difference(arr\_b, SIZE\_B, arr\_a, SIZE\_A, arr\_c, SIZE\_C);

assert(size\_c == 6);

printf("B Minus A:\t\t");

print\_int\_array(arr\_c, size\_c);

\*/

/\*

// 36.

#define SIZE 10

int arr[SIZE] = {0};

int arr\_size;

for(int i=0; i<SIZE; i++){

arr[i] = rand() % 10;

}

print\_int\_array(arr, SIZE);

arr\_size = remove\_duplicate\_in\_array(arr, SIZE);

print\_int\_array(arr, arr\_size);

assert(arr\_size == 7);

\*/

/\*

// 37.

#define SIZE 10

int arr[SIZE] = {0};

for(int i=0; i<SIZE; i++){

arr[i] = rand() % 10;

}

print\_int\_array(arr, SIZE);

assert(linear\_search(arr, SIZE, 4) == 2);

assert(linear\_search(arr, SIZE, 6) == -1);

\*/

/\*

// 38.

#define SIZE 10

int arr[SIZE] = {0};

for(int i=0; i<SIZE; i++){

arr[i] = rand() % 40;

}

assert(ascending\_sort(arr, SIZE));

print\_int\_array(arr, SIZE);

assert(binary\_search(arr, SIZE, 9) == 3);

assert(binary\_search(arr, SIZE, 6) == -1);

\*/

/\*

// 39.

#define SIZE 7

int arr[SIZE] = {4, 5, 2, 5, 6, 4, 7};

assert(sum\_of\_product(arr, SIZE) == 122);

\*/

/\*

// 40.

#define SIZE 100

char str[SIZE] = "FAST-TRACK C";

assert(find\_string\_length(str) == 12);

\*/

/\*

// 41.

#define SIZE 100

char str[SIZE];

worded\_date(17, 7, 1998, str);

assert(strcmp(str, "17th July 1999") == -1);

assert(strcmp(str, "17th July 1998") == 0);

\*/

/\*

// 42.

#define SIZE 100

char str[SIZE] = "Fast-Track C";

printf("%s\n", str);

lower\_to\_upper(str);

printf("%s\n", str);

\*/

/\*

// 43.

#define SIZE 100

char str[SIZE] = "Fast-Track C";

printf("%s\n", str);

reverse\_string(str);

printf("%s\n", str);

\*/

/\*

// 44.

assert(string\_palindrome("Fast-Track C") == 0);

assert(string\_palindrome("HiH") == 1);

assert(string\_palindrome("HiiH") == 1);

\*/

/\*

// 45.

#define SIZE 100

char str\_1[SIZE] = "Fast-Track ";

char str\_2[] = "C";

string\_concat(str\_1, str\_2);

assert(strcmp(str\_1, "Fast-Track C") == 0);

\*/

/\*

// 46.

#define ROW\_SIZE 5

#define COL\_SIZE 7

int \*\*matrix;

matrix = create\_2d\_matrix(ROW\_SIZE, COL\_SIZE);

assert(matrix != NULL);

assert(initialize\_2d\_matrix(matrix, ROW\_SIZE, COL\_SIZE) == 1);

assert(print\_2d\_matrix(matrix, ROW\_SIZE, COL\_SIZE) == 1);

\*/

/\*

// 47.

#define ROW\_SIZE 5

#define COL\_SIZE 5

int \*\*matrix;

matrix = create\_2d\_matrix(ROW\_SIZE, COL\_SIZE);

assert(matrix != NULL);

assert(initialize\_2d\_matrix(matrix, ROW\_SIZE, COL\_SIZE) == 1);

assert(print\_2d\_matrix(matrix, ROW\_SIZE, COL\_SIZE) == 1);

printf("\n");

assert(transpose\_2d\_matrix(matrix, ROW\_SIZE, COL\_SIZE) == 1);

assert(print\_2d\_matrix(matrix, ROW\_SIZE, COL\_SIZE) == 1);

\*/

/\*

// 48.

#define ROW\_SIZE 5

#define COL\_SIZE 5

int \*\*matrix;

matrix = create\_2d\_matrix(ROW\_SIZE, COL\_SIZE);

assert(matrix != NULL);

assert(initialize\_2d\_matrix(matrix, ROW\_SIZE, COL\_SIZE) == 1);

assert(print\_2d\_matrix(matrix, ROW\_SIZE, COL\_SIZE) == 1);

printf("\n");

assert(print\_column\_sum(matrix, ROW\_SIZE, COL\_SIZE) == 1);

\*/

/\*

// 49.

#define ROW\_SIZE 5

#define COL\_SIZE 5

int matrix\_1[ROW\_SIZE][COL\_SIZE] = {{2, 4, 6, 0, 9}, \

{1, 4, 6, 0, 9}, \

{2, 4, 6, 0, 9}, \

{8, 4, 6, 0, 9}, \

{2, 4, 6, 0, 9}};

int matrix\_2[ROW\_SIZE][COL\_SIZE] = {{2, 0, 6, 0, 0}, \

{0, 4, 0, 0, 9}, \

{2, 0, 6, 0, 9}, \

{0, 4, 0, 0, 9}, \

{2, 0, 6, 0, 0}};

assert(check\_if\_sparse(ROW\_SIZE, COL\_SIZE, matrix\_1) == 0);

assert(check\_if\_sparse(ROW\_SIZE, COL\_SIZE, matrix\_2) == 1);

\*/

/\*

// 50.

Student\_Array std\_arr;

// Assigning in this way is only possible during declaration

Student std\_1 = {8001, "Hrishikesh", 25, 7.97};

Student std\_2 = {8002, "Deon", 21, 8.3};

Student std\_3;

// Explicit assignment must be made after decleration

std\_3.roll\_no = 8003;

strcpy(std\_3.name, "Narendra");

std\_3.age = 24;

std\_3.marks = 8.5;

// Check initialize function

std\_arr = initialize\_array(3);

assert(std\_arr.c\_size == 0);

assert(std\_arr.t\_size == 3);

// Check store data function

std\_arr = store\_data(std\_arr, std\_1);

assert(std\_arr.c\_size == 1);

std\_arr = store\_data(std\_arr, std\_2);

std\_arr = store\_data(std\_arr, std\_3);

// Check print function

print\_data(std\_arr);

\*/

/\*

// 51.

Complex\_Num a = {4, 7};

Complex\_Num b = {0, 3};

Complex\_Num result;

// Test complex addition

result = add\_complex(a, b);

assert(result.real == 4);

assert(result.img == 10);

// Test complex subtraction

result = sub\_complex(a, b);

assert(result.real == 4);

assert(result.img == 4);

// Test complex multiplication

result = mul\_complex(a, b);

assert(result.real ==-21);

assert(result.img == 12);

\*/

/\*

// 57.

FILE \*fptr;

fptr = fopen("F:/C/Exploring-C/FastTrack-C/Practice-Problems/text\_file\_1.txt", "r");

assert(disp\_file\_content(fptr) == 1);

fclose(fptr);

\*/

/\*

// 58.

FILE \*dest, \*src;

src = fopen("F:/C/Exploring-C/FastTrack-C/Practice-Problems/text\_file\_1.txt", "r");

dest = fopen("F:/C/Exploring-C/FastTrack-C/Practice-Problems/text\_file\_2.txt", "w");

assert(copy\_file\_content(dest, src) == 1);

fclose(dest);

fclose(src);

\*/

/\*

// 59.

FILE \*fptr;

fptr = fopen("F:/C/Exploring-C/FastTrack-C/Practice-Problems/char\_count.txt", "r");

char\_count\_in\_file(fptr);

fclose(fptr);

\*/

/\*

// 60.

FILE \*num\_ptr;

FILE \*odd\_ptr;

FILE \*evn\_ptr;

num\_ptr = fopen("F:/C/Exploring-C/FastTrack-C/Practice-Problems/num\_file.txt", "r");

odd\_ptr = fopen("F:/C/Exploring-C/FastTrack-C/Practice-Problems/odd\_file.txt", "w");

evn\_ptr = fopen("F:/C/Exploring-C/FastTrack-C/Practice-Problems/evn\_file.txt", "w");

odd\_even\_sorting(num\_ptr, evn\_ptr, odd\_ptr);

fclose(num\_ptr);

fclose(odd\_ptr);

fclose(evn\_ptr);

\*/

// 61.

FILE \*fptr;

Contact contact\_1 = {"Hrishikesh", "Bharathi Complex, Manipal", "7349493697"};

Contact contact\_2 = {"Mother", "Iruvail, Moodbidri", "9902397988"};

Contact contact\_3 = {"Father", "Sheshadripuram, Bangalore", "8105895176"};

fptr = fopen("F:/C/Exploring-C/FastTrack-C/Practice-Problems/phone\_directory.txt", "r+");

// Test add record

assert(add\_record(fptr, &contact\_1));

assert(add\_record(fptr, &contact\_2));

assert(add\_record(fptr, &contact\_3));

fclose(fptr);

return 0;

}