TIC1001—Introduction to Computing and Programming
National University of Singapore

Midterm Test - AY17/18

Question 1: C/C++ Expressions

There are several parts to this question which are to be answered independently and separately. Each part consists of a fragment of C/C++ code. Write the exact output produced by the code in the answer box. If an error occurs, or it enters an infinite loop, state and explain why. You may show workings outside the answer box in the space beside the code. Partial marks may be awarded for workings if the final answer is wrong. Assume that all appropriate preprocessor directives e.g., #include < iostream > , etc. have already been defined.

In [1]:
```
#include <iostream>
using namespace std;
```

A.

In [ ]:
```
int square(double x)
{
    return x * x;
}
printf("%d", square(1.5));
```

0. Call `square(x) -> square(1.5)`

return 1.5 * 1.5 = 2.25

However return data type is int, return 2.

Print"2"

Ans: 2

B.

In [ ]:
```
int x = 2;
int square(int x)
{
    return x * x;
}
printf("%d", square(square(3)));
```

Initialize x = 2

0.Call `square(x) -> square(3)`

return 3 * 3 = 9, return 9

1.Call `square(x) -> square(9)`

return 9 * 9 = 81, return 81

Ans: 81

C.

In [ ]:
```
int x = 5;
int y = 3;
double foo(int x, int y)
{
    return x/y;
}
printf("%f", foo(y, x));
```

Initialize x = 5, y = 3

Call `foo(y,x) -> foo(3,5)` return 3 / 5 = 0

Print"0.000000"

D.

In [ ]:
```
int foo(int x, double y)
{
    printf("%.2f ", y/x);
    return y/x;
}
foo(foo(4, 10), 15);
```

0.Call `foo(4,10)`

10 / 4 = 2.50

Print"2.50", return 2

1.Call `foo(2,15)`

15 / 2 = 7.50

Print"7.50", return 7

Ans: 2.50 7.50

E.

In [ ]:
```
int i;
for (i = 0; i < 10; ++i)
{
    if (i % 2 == 0)
    {
        ++i;
    }
    if (i % 3 == 0)
    {
        --i;
    }
    if (i % 5 == 0)
    {
        break;
    }
}
printf("%d", i);
```

When i = 0,
0 % 2 = 0, will pass `if (i % 2 == 0)`
Increment i + 1 = 1
Continue to next If Statement
1 % 3 = 1, will not pass `if (i % 3 == 0)`
Continue to next If Statement
1 % 5 = 1, will not pass `if (i % 5 == 0)`
Continue to loop, increment i + 1, i = 2

When i = 2,
2 % 2 = 0, will pass `if (i % 2 == 0)`
Increment i + 1 = 3
Continue to next If Statement
3 % 3 = 0, will pass `if (i % 3 == 0)`
Decrement i - 1 = 2
Continue to next If Statement
2 % 5 = 2, will not pass `if (i % 5 == 0)`
Continue to loop, increment i + 1, i = 3

When i = 3,
3 % 2 = 1, will not pass `if (i % 2 == 0)`
Continue to next If Statement
3 % 3 = 0, will pass `if (i % 3 == 0)`
Decrement i - 1 = 2
Continue to next If Statement
2 % 5 = 2, will not pass `if (i % 5 == 0)`
Continue to loop, increment i + 1, i = 3

When i = 3,
3 % 2 = 1, will not pass `if (i % 2 == 0)`
Continue to next If Statement
3 % 3 = 0, will pass if (i % 3 == 0)
Decrement i - 1 = 2
Continue to next If Statement
2 % 5 = 2, will not pass `if (i % 5 == 0)`
Continue to loop, increment i + 1, i = 3

Ans: i will stuck at 3, infinite loop

F.

In [ ]:
```
int a = 3;
int f(int a)
{
    a = 7;
    return a;
}
int g(int a)
{
    return f(a) + a;
}
printf("%d", g(a+2) + f(a-1));
```

Initialize a = 3

Call `g(3+2) -> g(5)`
Call `f(a) ->` 7
Return 7 + 5 = 12

Call `f(a-1) -> f(2)`
Return 7

12 + 7 = 19,print"19"

G.

In [ ]:
```
inti=20,j=0;
while (i >= j)
{
    --i;
    if (i < 3*j)
    {
        continue;
    }
    j += 2;
    printf("%d ", j);
}
printf("%d", i);
```

Initialize i = 20, j = 0

20 >= 0, enter while loop.
Decrement i - 1, i = 19
19 > 3 * 0, will not pass `if (i < 3 * j)`
Increment j + 2, j = 2
Print"2 "

19 >= 2, enter while loop.
Decrement i - 1, i = 18
18 > 3 * 2, will not pass `if (i < 3 * j)`
Increment j + 2, j = 4
Print"4 "

18 >= 4, enter while loop.
Decrement i - 1, i = 17
17 > 3 * 4, will not pass `if (i < 3 * j)`
Increment j + 2, j = 6
Print"6"

17 >= 6, enter while loop.
Decrement i - 1, i = 16
16 < 3 * 6, will pass `if (i < 3 * j)`
Continue to loop

16 >= 6, enter while loop.
Decrement i - 1, i = 15
15 < 3 * 6, will pass `if (i < 3 * j)`
Continue to loop

15 >= 6, enter while loop.
Decrement i - 1, i = 14
14 < 3 * 6, will pass `if (i < 3 * j)`
Continue to loop

.....i will continue to decrement by 1 in every loop.

5 >= 6, will not enter while loop.
Print "5"

Ans: 2 4 6 5

H.

In [ ]:
```c
int x = 0;
int f(int *x)
{
    if (*x > 0)
    {
        *x = -*x;
    }
    else
    {
        *x += 2;
    }
    return *x;
}

x = f(&x) + f(&x);
printf("%d", x);
```

Initialize x = 0

0.Call `f(&x)`

* x = 0, will not pass <code>If (* x > 0)</code>. Enter Else statement
* x + 2 = 0 + 2 = 2, return 2

1.Call `f(&x)`

* x = 2, will pass <code>If (* x > 0)</code>.
* x = - 2, return -2

x = 2 + (-2) = 0
Print"0"

Ans: 0

Question 2: Traffic Lights

A traffic light has three states: green, amber and red. We can use an integer to represent the state of a traffic light using the values 0, 1, and 2, respectively.

A. Implement the function `void display(int light)` that takes as input a traffic light represented as an integer. The function prints the string `"Green"` , `"Amber"` or `"Red"` based on the value of the light.

In [ ]:
```c
void display(int light)
{
    if (light == 0)
    {
        printf("Green");
    }
    else if (light == 1)
    {
        printf("Amber");
    }
    else
    {
        printf("Red");
    }
}
```

B. In Singapore, our traffic lights cycles from green, to amber, to red, then back to green. The function `change` takes as input a traffic light (which is an integer) and changes it to the next state.
Decide on the appropriate type and inputs of the function `change` and provide an im- plementation of it in C.

In [ ]:
```c
void change(int *light)
{
    *light = (*light + 1) % 3;
}
```

C. We can consider one traffic light cycle to be a change from its current state to another, until it returns back to its starting state.
Suppose that a function `void wait(int secs)` has been defined which pauses the pro- gram execution for the given number of seconds. Implement a function `cycle` that takes as input a traffic light, and the timings between the states. The function changes and display the state of the traffic light with the appropriate pauses between the states for one cycle.

For example, if the light is currently green, the call `cycle(&light, 5, 20, 20)` will result in the following being printed (commented lines are not printed, but explains the behaviour):

Amber // *pause of 5 seconds*
Red // *pause of 20 seconds*
Green // *pause of 20 seconds*

Provide an implementation for the function `cycle` . Note that the value of the light has to change, and you should reuse the functions defined in the previous parts.

In [ ]:
```c
void cycle(int *light, int d1, int d2, int d3)
{
    change(light);
    display(*light);
    wait(d1);

    change(light);
    display(*light);
    wait(d2);

    change(light);
    display(*light);
    wait(d3);
}
```

D. A particular traffic light cycles with the following timings:

Cycle	Green	Amber	Red	Repeat
1	58s	2s	60s	10 times
2	38s	2s	90s	5 times
3	88s	2s	40s	5 times

Basically, cycle 1 will repeat 10 times, followed by cycle 2 for 5 times then cycle 3 for 5 times. Then the entire cycle will repeat from cycle 1 again.

Complete the following code such that the program will print the state of this particular traffic light when it changes with the required pause between changes. E.g., starting at cycle 1, `"Green"` will be printed followed by `"Amber"` after pausing for 58 s, and `"Red"` after a pause of 2 s. The program should continue running indefinately, displaying the changes according to the stated schedule.

You are to reuse the functions defined in the previous parts.

In [ ]:
```c
int main(void)
{
    int light = 0;
    int i;
    light = 2;

    while (1)
    {
    // loop forever
        // Cycle 1
        for (i = 0; i < 10; ++i)
        {
            cycle(&light, 58, 2, 60);
        }
        // Cycle 2
        for (i = 0; i < 5; ++i)
        {
            cycle(&light, 38, 2, 90);
        }
        // Cycle 3
        for (i = 0; i < 5; ++i)
        {
            cycle(&light, 88, 2, 40);
        }
    }
}
```

Question 3: Tic Tac Toe

Suppose we want to store the board state of a tic-tac-toe game, for example:
O | _ | O
O | X | _
X | _ | X
We can use numbers 0, 1 and 2 to represents the markers _ (empty space),O and × respectively. Then, going by rows from top to bottom, the whole board can be represented by a 9-digit integer T . For example, T = 101120202 represents the game board above.

A. Draw the tic-tac-toe board represented by the T = 202012111.

X | _ | X
_ | O | X
O | O | O

B. What is the smallest and largest possible value for T ? There is no need to consider whether the board can be the result of a actual game as long as each location has a valid marker.

Minimum = 0, Maximum = 222222222

C. The representation is not very efficient in terms of storage space (i.e. it uses more bits then absolutely necessary). Briefly explain why.

The full range of decimal value is not utilized.

D. Use your understanding of number base, briefly suggest a more efficient way to represent the game state as a single number.
Use your scheme to represent the example game board given in the main question. You can leave the representation in any number base as long as you indicate the base clearly.

Simply use base 3. The game board is $T = 101120202_3$

E. It turns out that the board represented in Part A is not a valid game. In a game of tic-tac-toe, the two players represented by O and ×, take alternating turns to fill an empty space with their mark. Assuming that × always goes first, there can never be more O than × and there can at most be 1 more × than O in a valid game board.

Implement the function `is_valid(int board)` which takes as input a game board using our representation given at the start of this question. The function returns `true` if the game board is valid based on the number of markers, and `false` otherwise.

Function Call	Return Value
is_valid(202012111)	false
is_valid(111222000)	true
is_valid(210120020)	true

In [ ]:
```c
#include <stdbool.h>

bool is_valid(int board)
{
    int x = 0, o = 0;
    while (board > 0)
    {
        if (board % 10 == 1)
        {
            ++o;
        }
        else if (board % 10 == 2)
        {
            ++x;
        }
        board = board / 10;
    }

    return (x == o || x-o == 1);
```

— END OF PAPER —