Tutorial 4

1. Explain the keyword: a) Block b) Boolen expression c) Loop

a) A block is created using a pair of curly braces. The block collects statements together into a single compound statements.

b) Boolean is a data type that contains two types of values, i.e., 0 and 1. The bool type value represents two types of behavior, either true or false. Here, '0' represents false value, while '1' represents true value.

c) Loop is used to execute the block of code several times according to the condition given in the loop.

2. What do you mean by control statement?

A control statement is a statement that determines whether other statements will be executed.

3. Explain the purpose of if-else statement?

The if/else statement executes a block of code if a specified condition is true. If the condition is false, another block of code can be executed. Use if to specify a block of code to be executed, if a specified condition is true. Use else to specify a block of code to be executed, if the same condition is false.

4. Explain the purpose of switch statement?

A switch statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each switch case.

Samuela Abigail
71762108039

5. What is the purpose of comma operator?

The comma operator (,) evaluates each of its operands (from left to right) and returns the value of the last operand. This lets you create a compound expression in which multiple expressions are evaluated, with the compound expression's final value being the value of the rightmost of its member expressions.

6. Compare between a pre-test and post test loop?

A pre test loop tests its condition before each iteration. A post test loop tests its condition after each iteration. A post test loop will always execute at least once.

7. Mention the reason " why is the use of goto statement discouraged?

goto statement is highly discouraged in any programming language because it makes difficult to trace the control flow of a program, making the program hard to understand and hard to modify.

8.Is the relational expression a < b < c legal in C?

No it's not legal in C since only one operand condition can be evaluated at a time.

9. There is no logical exclusive OR operator in c program, can it it be simulated?

Logical XOR is the same as logical "not equal to." So just use != with Boolean values to stimulate OR operator in C.

Samuela Abigail
71762108039

10. The floating point numbers are seldom equals to required value of a variable, then how can floating point values or variables be tested for equality?

To compare two floating-point numbers then rather than using "==" operator we will find the absolute difference between the numbers (which if were correctly represented, the difference would have been 0) and compare it with a very small number 1e-9 (i.e 10^-9, this number is very small) and if the difference is less than this number, we can safely say that the two floating-point numbers are equal.

11. Explain null statement?

It is an expression statement with the expression missing. It is useful when the syntax of the language calls for a statement but no expression evaluation. It consists of a semicolon.

12. Which form of loop should use – while or do-while?

Both while and do-while loop are the iteration statement, and it's usage depends on the situation. If we want the given condition to be verified first, and then the statements inside the loop must execute, then the while loop is used. If we want to test the termination condition at the end of the loop, then the do-while loop is used.

13. Differentiate between break and continue statement

Break statement-

- It is used to exit from the loop constructs.
- When a break statement is encountered then the control is exited from the loop construct immediately.

Continue statement-

- The continue statement is not used to exit from the loop constructs.
- When the continue statement is encountered then the control automatically passed from the beginning of the loop statement.

14. Is there any difference between the following for statement? Explain?

a) for(x = 1; x < 100; x++)

b) for(x =1; x < 100; ++x)

c) for (x = 1; x < 100; x = x + 1)

d) for (x = 1; x< 100; x += 1)

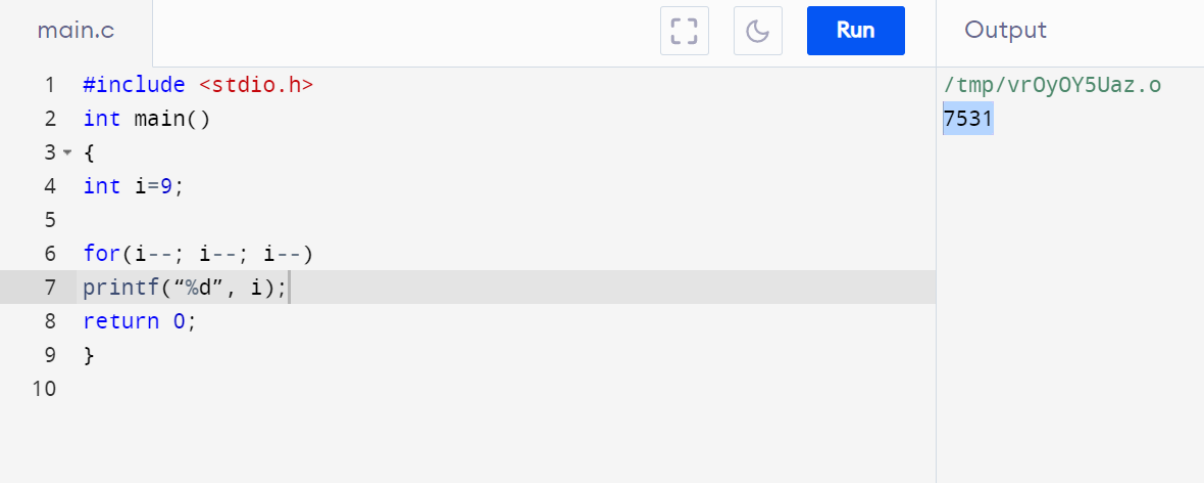
Option (a) and (b) will give same output only, but in the former the increment will be read after reading the variable x, whereas in the latter the increment operand will be read before x by the compiler. Options (c) and (d) will also give the same output of (a) and (b) only, except that the condition in (c) is x=x+1 and in (d) it's x+=1

15. What would be the output of following program?

```
int main()
{
int i=9;
for(i--; i--; i--)
printf("%d", i)
return 0;
```

Samuela Abigail
71762108039

}



16. What will be printed by the code below?

a)

float x = 123.4

If (x< 100)

printf ("one ");

If (x < 200)

printf (" two ");

If (x < 300)

printf(" three ");

main.c

Run

Output

```c
#include <stdio.h>
int main()
{
float x = 123.4;

if ( x< 100)
{printf ("one");}
if ( x < 200)
{printf ( "two");}
if (x < 300)
{printf( "three");}
return 0;
}
```

```
/tmp/vrOyOY5Uaz.o
twothree
```

b)

char c = ‘y’;

switch(c)

{

case ‘Y’; printf(“Yes/No”);

break;

case ‘N’; printf(“No/ Yes”);

break;

default: printf(“other”);

}

Samuela Abigail
71762108039

main.c

Run

Output

```c
#include <stdio.h>
int main()
{
char c = 'y';
switch(c)
{
case 'Y': printf("Yes/No");
break;
case 'N': printf("No/ Yes");
break;
default: printf("other");
}

return 0;
}

```

```
/tmp/vrOyOY5Uaz.o
other
```

Try the below project-based questions:

1. Write a C program that prompts the user to enter the date as three integer values for the month, the day in the month and year. The program should then output the data in the form of 31 st December 2003 when the user enters 12 31 2010, say. The program has to work out when c “th”, “nd”, “st”, and “rd” need to be appended to the day value. The programmer should not forget 1 St , 2 nd , 3 rd , 4 th ; and then 11 th , 12 th , 13 th ,14 th ; and 21 st , 22 nd , 23 rd , and 24 th .

```
#include <stdio.h>

#include <string.h>


int main() {

   int day, month, year, n;

   char
mn1[15]="January",mn2[15]="February",mn3[15]="March",mn4[15]
="April"
```

Samuela Abigail
71762108039

```
,mn5[15]="May",mn6[15]="June",mn7[15]="July",mn8[15]="August"
,mn9[15]="September",mn10[15]="October",mn11[15]="November"
,mn12[15]="December";

   printf("Enter day: ");
   scanf("%d", &day);

   printf("Enter month: ");
   scanf("%d",&month);

   printf("Enter year: ");
   scanf("%d",&year);

if (day >10 && day<14 )
{
   n = day;
}
else {
   n=day %10;
}

   switch(n)
   {
      case 1:
```

```
        printf("Date is %dst ", day);
        break;

    case 2:
        printf("Date is %dnd ", day);
        break;

    case 3:
        printf("Date is %drd ", day);
        break;

    default:
        printf("Date is %dth ", day);
        break;
  }

switch(month)
  {
    case 1:
        printf("%s, %d\n",mn1, year);
        break;

    case 2:
        printf("%s, %d\n",mn2, year);
```

```
        break;

    case 3:
        printf("%s, %d\n",mn3, year);
        break;

    case 4:
        printf("%s, %d\n",mn4, year);
        break;

    case 5:
        printf("%s, %d\n",mn5, year);
        break;

    case 6:
        printf("%s, %d\n",mn6, year);
        break;

    case 7:
        printf("%s, %d\n",mn7, year);
        break;

    case 8:
        printf("%s, %d\n",mn8, year);
```

```
        break;

    case 9:
        printf("%s, %d\n",mn9, year);
        break;

    case 10:
        printf("%s, %d\n",mn10, year);
        break;

    case 11:
        printf("%s, %d\n",mn11, year);
        break;

    case 12:
        printf("%s, %d\n",mn12, year);
        break;

    default:
            printf("Invalid month number");
            break;
  }

  return 0;
```

Samuela Abigail
71762108039

}

main.c

Run

Output

```c
#include <stdio.h>
#include <string.h>

int main() {
    int day, month, year, n;
    char mn1[15]="January",mn2[15]="February",mn3[15]="March"
        ,mn4[15]="April" ,mn5[15]="May",mn6[15]="June",mn7[15]
        ="July",mn8[15]="August",mn9[15]="September",mn10[15]
        ="October",mn11[15]="November",mn12[15]="December";

    printf("Enter day: ");
    scanf("%d", &day);

    printf("Enter month: ");
    scanf("%d",&month);

    printf("Enter year: ");
    scanf("%d",&year);

```

```
/tmp/DwUFietoTa.o
Enter day: 19
Enter month: 03
Enter year: 2004
Date is 19th March, 2004
```

2. This is a well-known game with number of variants. The following variants has an interesting winning strategy. Two players alternatively take marbles from a pile. In each move, a player chooses how many marbles to take. The player must take at least one but at most half of the marbles. Then the other player takes a turn. The player who takes the last marble loses. Write a C program in which computer plays against a human opponent. Generate a random integer between 10 and 100 to denote the initial size of the pile. Generate a random integer between 0 and 1 to decide whether the computer or human takes the first turn, generate a random integer between 0 and 1 to decide whether computer play smart or stupid. In the stupid mode, the computer simply takes a random legal value (between 1 and n/2, where n is the total number of marbles) from the piles whenever it has a turn. In smart mode, the computer takes off enough marbles to make the size of the piles a power of two minus- that is, 3, 7, 15, 31 or 63. That is always a legal move, except when the size of pile is currently one less than power of two. In that case, the computer makes a random legal move. It

should be noted that the computer cannot be beaten in smart mode when it has the first move, unless the pile size happens to be 15, 31 or 63. Of course a human player who has the first turn and knows the winning strategy can win against the computer.

Samuela Abigail
71762108039