Tut. Sheet 1

Solutions @AyushR1

1. Which of the following are invalid variable names and why?

`B'day` Invalid: No special symbols are allowed.

`int` Invalid: keywords cannot be used as a variable name

`$hello` valid:.

`#HASH` Invalid: No special symbols are allowed.

`dot.` Invalid: No special symbols are allowed.

`number` Valid

`totalArea` Valid

`_main()` Invalid: No special symbols are allowed.

`temp_in_Deg` Valid: _ is an exception

`total%` Invalid: No special symbols are allowed.

`1st` Invalid: Variable name cannot start with a number.|

`stack-queue` Invalid: No special symbols are allowed.

2. Point out the errors, if any, in the following C statements:

a.

```
char = ‘3’ ;
```

Ans :-

The variable name is not defined

only single sided single quotes are used to enclose char

b.

```
4 / 3 * 3.14 * r * r * r = vol_of_sphere ;
```

Ans :-

One and only one variable can be at the left side of an expression.

c.

```
volume = a^3 ;
```

Ans :-

There is no exponential operator in C.

We can use pow(x,n) where x is the base and n is power.

d.

```
area of circle = 3.14 * r * r ;
```

Ans :-

No spaces are allowed in c variable names.

e.

```
slope = ( y2 – y1 ) ÷ ( x2 – x1 ) ;
```

Ans

wrong divide symbol

f.

```
char ch = '25 Apr 12' ;
```

Ans

Char can only hold a single character.

g.

```
si = p * r * n / 100 ;
```

Ans.

No error

3. Convert the following algebraic expressions into equivalent C statements:

a) $A = \frac{7.7b(xy+a)/c-0.8+2b}{(x+a)(1/y)}$

b) $R = \frac{2v+6.22(c+d)}{g+v}$

a. (((7.7 * b (x * y + a)) / c)- 0.8 + 2 * b)/ ((x + a)*(1 / y));

b. (2 * v + 6.22 * (c + d)) / (g + v);

4. What will be the output of the following programs?

```
#include <stdio.h>

int main ()
{
  int i = 2, j = 3, k, l;
  float a, b;

  k = i / j * j;

  l = j / i * i;

  a = i / j * j;

  b = j / i * i;
  printf ("%d %d %f %f\n", k, l, a, b);
  return 0;
}
```

Ans :-

0 2 0.000000 2.000000

```
/* Calculation of average
/* Author: Sanjay */
/* Place – Whispering Bytes */
*/
#include <stdio.h>
int main( )
{
int a = 35 ; float b = 3.24 ;
printf ( "%d %f %d", a, b + 1.5, 235 ) ;
}
```

Ans ;-

35 4.740000 235

```
# include <stdio.h>
int main( )
{
```

```
int a, b ;
printf ( "Enter values of a and b" ) ;
scanf ( " %d %d ", &a, &b ) ;
printf ( "a = %d b = %d", a, b ) ;
return 0 ;
}
```

Ans

Depend on input

eg. inputs 7 8

a = 7 b = 8

5. Describe the difference between the literal values 7, "7", and ’7’ ?

Ans:

The first literal is integer 7.

Second literal is null terminated string value '7'.

Third literal is character '7' having ASCII character code (55).

6. Consider the statement

```
double ans = 10.0+2.0/3.0−2.0∗2.0;
```

Rewrite this statement, inserting parentheses to ensure that ans = 11.0 upon evaluation of this statement ?

Answer:

double ans = 10.0+2.0/((3.0−2.0)∗2.0);

7. Consider the statement

```
double ans = 18.0/squared(2+1);
```

For each of the four versions of the function macro squared() below, write the corresponding value of ans.

1. #define squared(x) x*x

2. #define squared(x) (x*x)

3. #define squared(x) (x)*(x)

4. #define squared(x) ((x)*(x))

1. **#define squared(x) x*x**

18.0/ 2+1 * 2+1

18.0/2+2+1

9+2+1

double ans= 12

2. **#define squared(x) (x*x)**

18.0/ (2+1 * 2+1)

18.0/(2+2+1)

18.0/5

double ans = 3.6

3.**#define squared(x) (x)*(x)**

18.0/ (2+1) *(2+1)

18.0/3*3

6*3

double ans = 18

4.**#define squared(x) ((x)*(x))**

18.0/ ((2+1) *(2+1))

18.0/(3*3)

18.0/9

double ans = 2

8. The following lines of code, when arranged in the proper sequence, output the simple message “All your base are belong to us.”

1. return 0;
2. const char msg[] = MSG1;
3. }
4. #define MSG1 "All your base are belong to us!"
5. int main(void) {
6. #include <stdio.h>
7. puts(msg);

```
#include <stdio.h>

#define MSG1 "All your base are belong to us!"

int main(void) {

    const char msg[] = MSG1

    puts(msg)
```

```
    return 0
}
```

9. For each of the following statements, explain why it is not correct, and fix it.

a.

```
#include <stdio.h>;
```

b.

```
int function(void arg1)
    {
        return arg1-1;
      }
```

c.

```
#define MESSAGE = "Happy new year!"

puts(MESSAGE);
```

Ans :-

a. no semi colon

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

1. arg1 is of type void

```
int function(int arg1)
    {
        return arg1-1;
      }
```

1. no = in define

```
#define MESSAGE "Happy new year!"

puts(MESSAGE);
```

4. float f = 4.3471376;

Then what will the output of

a) printf("%f\n",f); 06

b) printf("%.2f\n",f);

c) printf("%.3f\n",f);

d) printf("%2f\n",f);

e) printf("%5.2f\n",f);

f) printf("%8.10f\n",f);

```
4.347137
4.35
4.347
4.347137
' 4.35'
4.3471374512
```