1. Explain the effect of the following code:

```
int i;
…
while (i = 2) {
    printf (“Examples of even numbers are: %d %d %d\n”, i, i+2, i+4);
    i = 0;
}
```

Where is the source of the problem?

Answer:
Infinite loop that keeps printing “Examples of even numbers are: 2 4 6”. This is because the while-loop condition is an assignment expression instead of to test, i.e. i==2.

2. In the following code to calculate the area of a circle using a function, will it compile properly? If not, how do you solve the problem? Explain your solution.

```
#include <stdio.h>

float AreaOfCircle(float);

int main(void)
{
    float radius, area;
    float PI = 22/7.0;

    printf("Radius = ");
    scanf("%f", &radius);
    area = AreaOfCircle (radius);
    printf("Area = %f\n", area);
    return 0;
}

float AreaOfCircle (float r)
{
    return (PI * r * r); /* area equals Pi times radius squared*/
}
```

Answer:
Compilation Error – PI is not visible to the function AreaOfCircle;

Solution: move declaration “float PI = 22/7.0” out of main() function.

3. What is the difference between the declaration and definition of a data object or function? You can use the code example in Q2 to explain.

Answer:
A *declaration* announces the properties of a data object or a function. The main reason for declaring data objects and functions is type checking. If a variable or function is declared and later reference is made to it with data objects that do not match the types in the declaration, the compiler will complain. The purpose of the complaint is to catch type errors at compile time rather than waiting until the program is run, when the results can be more fatal.

A *definition*, on the other hand, actually sets aside storage space (in the case of a data object) or indicates the sequence of statements to be carried out (in the case of a function).

4. Which of the following is an **incorrect** assignment statement? Explain briefly.
   (a) n = m = 0
   (b) value += 10
   (c) mySize = x < y ? 9 : 11
   (d) testVal = (x > 5 || x < 0)

(e) none of the above

All are valid expressions for an assignment statement.

5. When you compile and execute the following code, what will be the output? Explain.

```
#include <stdio.h>

int main() {
        float c= 3.14;
        printf("%f", c%2);
        return 0;
}
```

Answer:
Output – Compiler error
Explanation – The % operator is applied on a variable of type float. The operands of the % opearator cannot be of float or double. This is why it causes a compiler error.

6. What is the output when you execute the following code? Explain.

```
#include <stdio.h>

int main() {
    int a=5;
    a=printf("Good")+ printf("Boy");
    printf("%d",a);
    return 0;
}
```

Answers:

Output – GoodBoy7

Explanation – printf() function returns the number of characters printed on the screen. 'Good' and 'Boy' will be printed consecutively. The first printf() returns 4 and the second one returns 3. So 4 + 3 = 7 is stored in a. When it is printed, 7 would be printed at the end of 'GoodBoy'.