Computer Science & Engineering Department
I. I. T. Kharagpur

Software Engineering: CS20006

Assignment – 1: Better C & Guidelines *Marks: 30*

Assign Date: *19th January, 2021* Submit Date: *23:55, 23rd January, 2021*

Instructions: Please solve the questions using pen and paper and scan the images. Every image should contain your roll number and name.

1. Consider the following program which should obviously print `Match`.

```
#include <iostream>
#include <cmath>
using namespace std;

#define sqr(x) ((x) * (x))

int main() {
    double a = 4.0*atan(1.0); // pi
    double b = sqrt(a); // square-root of pi

    if (a == sqr(b)) // pi is equal to
                     // square of square-root of pi?
        cout << "Match" << endl;
    else
        cout << "Mis-Match" << endl;

    return 0;
}
```

However, on Visual C++ 64-bit compiler, it prints `Mis-Match`. Identify the bug and fix it. **[1 + 1 = 2]**

Write an appropriate guideline to avoid such bugs and improve the quality of the code. **[1]**

2. What is the output of the following program? **[1]**

```
#include <iostream>
using namespace std;

int main() {
    int *p = new int(5);

    if (p = 0)
        cout << "No Value" << endl;
    else
        cout << *p << endl;

    return 0;
}
```

Is it what the developer intended? If yes, justify the thoughts of the developer. If no, find the bug in the program and fix it. **[1]**

Write an appropriate guideline to avoid such bugs and improve the quality of the code. **[1]**

3. Consider the following program:

```
#include <iostream>
using namespace std;

int rem(int n, int r) {
    return n % r;
}

int main() {
    int n = 15, r = 0; // Line 1

//  int n, r; // Line 2
//  cin >> n >> r; // Line 3

    if (r == 0 || rem(n, r))
        cout << "True" << endl;
    else
        cout << "False" << endl;

    if (rem(n, r) || r == 0)
        cout << "True" << endl;
    else
        cout << "False" << endl;

    return 0;
}
```

While using Visual C++ 64-bit compiler, the output is as follows:

Build Type	**Output**
Debug (Un-optimized)	`True` `Un-handled Floating Point Exception`
Release (Optimized)	`True` `True`

Now let us comment `Line 1` and un-comment `Line 2` & `Line 3`. The output changes to the following while we input `15` for `n` and `0` for `r` (as was initialized in `Line 1`:

Build Type	**Output**
Debug (Un-optimized)	`True` `Un-handled Floating Point Exception`
Release (Optimized)	`True` `Un-handled Floating Point Exception`

Explain the behavior in both cases, especially justifying the difference due to changing `Line 1` to `Line 2` & `Line 3` and providing the same input. **[2 + 2 = 4]**

Write an appropriate guideline to avoid such bugs and improve the quality of the code. **[1]**

4. Consider the following program having 6 functions - each being a slight variant of the other. State the behavior (like *compilation error*, *wrong output*, *run-time exception*, *correct output - showing the output*, *unpredictable behavior*, etc.) of each function with proper justification (refer to specific lines in a function as you may need) of the behaviour as stated. You may compare the functions also from the perspectives of the quality of the code. Make a table in the following format in your submission sheet and fill up accordingly. **[0.3 * 6 = 3]**

Function Name	Behaviour	Justification & Comments
f1()		
f2()		
...		
f6()		

Finally, based on the observations above, formulate guidelines to maintain a good quality of code. **[2]**

```
#include <iostream>
#include <cstring>
using namespace std;

void f1() {
    char * str = "Bat";
    cout << str << endl;
    str[0] = 'C';
    cout << str << endl;
    str = "Rat";
    cout << str << endl;
    cout << endl;
}

void f2() {
    const char * str = "Bat";
    cout << str << endl;
    str[0] = 'C';
    cout << str << endl;
    str = "Rat";
    cout << str << endl;
    cout << endl;
}

void f3() {
    char * const str = "Bat";
    cout << str << endl;
    str[0] = 'C';
    cout << str << endl;
    str = "Rat";
    cout << str << endl;
    cout << endl;
}

void f4() {
    char * str = strdup("Bat");
    cout << str << endl;
    str[0] = 'C';
    cout << str << endl;
    str = strdup("Rat");
    cout << str << endl;
    cout << endl;
}

void f5() {
    const char * str = strdup("Bat");
    cout << str << endl;
    str[0] = 'C';
    cout << str << endl;
    str = strdup("Rat");
    cout << str << endl;
    cout << endl;
}
```

```
void f6() {
    char * const str = strdup("Bat");
    cout << str << endl;
    str[0] = 'C';
    cout << str << endl;
    str = strdup("Rat");
    cout << str << endl;
    cout << endl;
}

int main() {
    f1();
    f2();
    f3();
    f4();
    f5();
    f(6);

    return 0;
}
```

5. Consider the following program where 24 lines have been marked. State the behavior (like *compilation error*, *wrong output*, *run-time exception*, *correct output - showing the output*, *unpredictable behavior*, etc.) of each line with proper justification (refer to specific lines in a function as you may need) of the behaviour as stated. Make a table in the following format in your submission sheet and fill up accordingly.

[0.5 * 24 = 12]

Function Name	**Behaviour**	**Justification & Comments**
Line 01		
Line 02		
...		
Line 24		

Finally, based on the observations above, formulate guidelines to maintain a good quality of code. **[2]**

```
#include <iostream>
#include <cmath>
using namespace std;

int e(int x) {
    cout << "x = " << x << " &x = " << &x << endl;
    return (x);
}

int f(int &x) {
    cout << "x = " << x << " &x = " << &x << endl;
    return (x);
}
int& g(int x) {
    cout << "x = " << x << " &x = " << &x << endl;
    return (x);
}
int& h(int &x) {
    cout << "x = " << x << " &x = " << &x << endl;
    return (x);
}
```

```
int main() {
    int a = 10;
    cout << "a = " << a << " &a = " << &a << endl;

    int& rvv = e(a); // Line 01
    int& rrv = f(a); // Line 02
    int& rvr = g(a); // Line 03
    int& rrr = h(a); // Line 04

    cout << "rvv = " << rvv << " &rvv = " << &rvv << endl; // Line 05
    cout << "rrv = " << rrv << " &rrv = " << &rrv << endl; // Line 06
    cout << "rvr = " << rvr << " &rvr = " << &rvr << endl; // Line 07
    cout << "rrr = " << rrr << " &rrr = " << &rrr << endl; // Line 08

    const int& rvvc = e(a); // Line 09
    const int& rrvc = f(a); // Line 10
    const int& rvrc = g(a); // Line 11
    const int& rrrc = h(a); // Line 12

    cout << "rvvc = " << rvvc << " &rvvc = " << &rvvc << endl; // Line 13
    cout << "rrvc = " << rrvc << " &rrvc = " << &rrvc << endl; // Line 14
    cout << "rvrc = " << rvrc << " &rvrc = " << &rvrc << endl; // Line 15
    cout << "rrrc = " << rrrc << " &rrrc = " << &rrrc << endl; // Line 16

    e(a) = 1; // Line 17
    cout << "a = " << a << " &a = " << &a << endl; // Line 18
    f(a) = 2; // Line 19
    cout << "a = " << a << " &a = " << &a << endl; // Line 20
    g(a) = 3; // Line 21
    cout << "a = " << a << " &a = " << &a << endl; // Line 22
    h(a) = 4; // Line 23
    cout << "a = " << a << " &a = " << &a << endl; // Line 24

    return 0;
}
```