Programming in C++: Assignment Week 2

Total Marks : 20
Each question carries one mark
Right hand side of each question shows its Type (MCQ/MSQ)

March 3, 2017

Question 1

- Look at the code snippet below:

```
int * const p = &n;
```

Which of the following statement is true for the variable 'p'? *Mark 1*

a. const-Pointer to non-const-Pointee
b. non-const-Pointer to const-Pointee
c. const-Pointer to const-Pointee
d. non-const-Pointer to non-const-Pointee
Answer: a
Explanation: As per syntax, refer slides

Question 2

- Look at the following code segment and decide which statement(s) is/are correct. *Mark 1*

```
int main(){
    int m = 4;
    const int n = 5;
    const int * p = &n;
    int * const q = &m;
    // ...
    n = 6;  // stmt-1
    *p = 7; // stmt-2
    p = &m; // stmt-3
    *q = 8; // stmt-4
    q = &n; // stmt-5
    // ...
}
```

a. stmt-1
b. stmt-2
c. stmt-3
d. stmt-4

e. stmt-5
Answer: c, d
Explanation: As per syntax, refer slides

Question 3

- Identify the output of the following code. *Mark 1 Mark 1*

```
#include<iostream>
using namespace std;
int main() {

    typedef struct Complex {
        double re;
        double im;
    } Complex;
   const Complex c = {2,4} ;
   c.re = 5.9;
   cout << c.re;
   return 0;
}
```

a. 5.9
b. Cannot assign an integer value to a double variable
c. 5.90
d. Cannot assign value 5.9 to read only c.re
Answer: d
Explanation: c is variable of the structure Complex, but it is defined as const, hence cannot be modified

Question 4

- Identify the correct statement(s). *Mark 1*

```
#include <iostream>
#include <cmath>
using namespace std;
#define TWO 2
#define PI 4.0*atan(1.0)
int main() {
    int r = 10;
    double peri = TWO * PI * r;
    cout << "Perimeter = "
    << peri << endl;
    return 0;
}
```

a. TWO and PI are variables
b. Types of TWO and PI may be indeterminate
c. Types of TWO and PI are determinate

d. TWO and PI look like variables
Answer: b), d)
Explanation: TWO and PI are manifest constants, hence types can be indeterminate and look like variables.

Question 5

- What will be the output of the following code? *Mark 1*

```
#include <iostream>
using namespace std;
double Ref_const(const double &param) {
    return (param * 3.14);
}
int main() {
    double x = 8, y;
    y = Ref_const(x);
    cout << x  << " "<< y;
    return 0;
}
```

a. Cannot return constant parameter

b. Cannot edit constant parameter

c. 8 26

d. 8 25.12
Answer: d)
Explanation: Const used to pass reference parameter param to prevent from being modified. The value of param is used only

Question 6

- What will be the output of the following code? *Mark 1*

```
#include <iostream>
using namespace std;
void func(int n1 = 10, int n2) {
    cout <<n1 << "  "<< n2;
}
int main() {
   func(1);
   func(3, 4);
   return 0;
}
```

a. 1 10 3 4

b. 10 1 4 3

c. 10 1 3 4

d. Compilation error: Argument missing for parameter 2 of func
Answer: d)
Explanation: Default values needs to specified from the end, hence function resoultion fails

Question 7

- What will be the output of the following code? *Mark 1*

```
#include <iostream>
using namespace std;
int Add(int a, int b) { return (a + b); }
double Add(double c) {
    return (c + 1);
}
int main() {
   int x = 1, y = 2, z;
   z = Add(x, y);

   cout <<  z;
   double s = 4.5, u;
   u = Add(s);

   cout << "  " << u << endl;
   return 0;
}
```

a. Add cannot be overloaded with different return types

b. Add cannot be resolved

c. 3 5.5

d. 3 6.5
Answer: c)
Explanation: Two versions of function Add called as per resolution

Question 8

- Which function prototype will match the function call *func(3.6,7)*? *Mark 1*

```
void func(int, int); // Proto 1
void func(double, double, double = 5.6); // Proto 2
void func(double, double, char = 'c'); // Proto 3
void func(double, char = 'd', char = 'c'); // Proto 4
```

a. Proto 1

b. Proto 2

c. Proto 3

d. Proto 4
Answer: a), b), c)
Explanation: Proto 1 allowed, as 3.6(1st parameter) is downcast to integer. Proto 2 allowed, as default value will be used for third parameter. Proto 3 allowed, default value and type will be used for third parameter. Proto 4 fils for mismatch in 2nd parameter

Question 9

- What will be the output of the following code? *Mark 1*

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

int main() {
    int *ptr = NULL;
    cout << " Output: In Program";
    delete ptr;
    return 0;
}
```

a. ptr cannot point to NULL

b. delete ptr (NULL) causes program crash

c. Output: In Program

d. Invalid Syntax

Answer: c)

Explanation: Null assignment to pointer allowed. Normal print provides the output

Question 10

- Fill up the blanks to get the desired output according to the test cases. *Mark 1*

```
#include <iostream>
#include <cstring>
#include <cstdlib>
using namespace std;
typedef struct _String { char *str; } String;
---------------------------------- {

    String s;
    s.str = (char *) malloc(strlen(s1.str) +
    strlen(s2.str) + 1);
    strcpy(s.str, s1.str);
    strcat(s.str, s2.str);
    return s;
}
int main() {
   String s1, s2, s3;
   s1.str = strdup("I");
   s2.str = strdup(" love Travelling ");
   s3 = s1 + s2;

   cout << s3.str << endl;
   return 0;
}
```

a. String + operator(const String& s1, const String& s2)

b. String +(const String& s1, const String& s2)

c. String operator+(const String& s1, const String& s2)

d. string operator+(const String s1, const String& s2)
Answer: c)
Explanation: As per syntax, Overloading operator + for String structure. Reference parameters passed as const to prevent modification.

I Programming Assignments

Question 1

- Fill up the blanks by providing appropriate return type and argument type for the function *Ref_func()* to get the desired output according to the test cases. *Marks 2*

```
#include <iostream>
using namespace std;
_____ Ref_func( _____ param) {

     return (++param);
}
int main() {
    int x, y, z;
    cin >> x ;
    cin >> y ;
    y = Ref_func(x);
    cout << x  << " "<< y << endl;

    Ref_func(x) = z;
    cout << x  << " "<< y;
    return 0;
}
```

Input: 8, -9
Output: 9 9
-9 9
Input: 10, 20
Output: 11 11
20 11
Input: -19, -32
Output: -18 -18
-32 -18

Answer: //int& // int &

Question 2

- Fill up the blanks to get the desired output according to the test cases. *Marks 2*

```
#include <stdio.h>
int func(int, int);
#define func(x, y) __ / __ + ___  // Complete the Macro definition
int main() {
    int i,j;

    scanf("%d", &i);
    scanf("%d", &j);
    printf("%d ",func(i + j, 3));
    ______ func   // Fill the blank
    printf("%d\n",func(i + j, 3));
}
int func(int x, int y) {
    return x / y + x;
}
```

a. Input: -6, 3 Output: -8 -4

b. Input: 11, 15 Output: 42 34

c. Input: -4, -8 Output: -18 -16

Answer: #define func(x, y) x / y + x
#undef func

Question 3

- Fill up the blanks with appropriate keyword to get the desired output according to the test cases. *Marks 2*

```
#include <iostream>
using namespace std;
______ int SQUARE(int x) { ______ x * x; }
int main() {
    int a , b, c;

    cin >> a ;
    b = SQUARE(a);
    cout << "Square = " << b << ", ";
    c = SQUARE(++a);
    cout << "++ Square = " << c ;
    return 0;
}
```

a. Input: 4 Output:Square = 16, ++ Square = 25

b. Input: -8 Output: Square = 64, ++ Square = 49

c. Input: -10.5 Output: Square = 100, ++ Square = 81

Answer: inline // return

Question 4

- Fill up the blanks to get the desired output according to the test cases in the perspective of dynamic memory allocation and de-allocation. *Marks 2*

```
#include <iostream>
using namespace std;
int main(){
    int d;
  ____________________ // use operator new to allocate memory to variable 'p'

    cin>> d ;
    *p = d ;
    cout << ++*p + d++ * (++*p + *p);
    __________delete(___); // Fill the blank
    return 0;
}
```

a. Input: -7 Output: 64

b. Input: 11.5 Output: 298

c. Input: 15 Output: 526

Answer: int *p = (int *)operator new(sizeof(int));// operator // p

Question 5

- Overload the function 'Area', by writing the appropriate definition in place of blank to get the desired output according to the test cases. *Marks 2*

```
include<iostream>
using namespace std;
___________________________ // Overload the function 'Area'
___________________________ // Write the definition of 'Area'
int main() {
    int x ,y, t;
    double z, u, f;

    cin >> x >> y ;
    cin >> z >> u ;

    t = Area(x);
    cout << "Area = " << t << " ";
    f = Area(z);
    cout << "Area = " << f << " ";
    f = Area(z,u);
    cout << "Area = " << f ;
    return 0;
}
```

a. Input: 8, 7, 9, 10 Output: Area =80 Area =90 Area =90

b. Input: 8, 9, 8.5, 9.5 Output: Area =80 Area =80 Area =80.75

c. Input: 7, 8, 7, 9.6 Output: Area =70 Area =70 Area =67.2

Answer: int Area(int a, int b = 10) { return (a * b); } //
double Area(double c, double d) { return (c * d); }