Department of Electrical and Computer Engineering
Concordia University

Programming Methodology I - COEN 243

Fall 1999 - Sections: U and W

Quiz #1 (Sunday Oct. 17, 1999)

Solve all questions
Time: 90 minutes

Total Marks = 15 points
Closed book.

Question 1. What is logically wrong with the following code: Correct it,

```
if (total = = MAX)
        if (total < sum )
          cout << " total equals MAX and total is less than sum ";
else
          cout << " total is not equal to MAX ";
```

(2 points)

Question 2. What is the value of the integer variables *total* and *num* after each of the following statements? Assume the variables *total* and *num* contain the values 2 and 3, respectively at the beginning of each statement.

a. total = ++ num; total 4 , num = 4
b. num = total ++; num : 2 , total = 3

(1 points)

Question 3. Write a program which has the same effect as the program below but which uses a while loop instead of do-while loop.

```
M=2;
S=0;
I=0;
do
{       S=S+M;
        M=3*M+I;
        I++
        cout << "M=" << M << endl;
} while (M<100);
```

```
while (M< 100)
  { S = S+M ;
    M = 3*M+I ;
    I++ ;
    cout << "M = " << M << endl;
  }
```

(2 points)

Question 4. Rewrite the following *switch* statement using *if-else* statement,

```
switch (number) {

        case 1:

        case 2:
              cout << " digits ";
        break;

        case 10:
                  cout << " tens " ;
        break;
```

```
case 100:
        cout << " hundreds ";
 break;

case 10000:
        cout << " ten ";

 case 1000:
        cout  << " thousands ";
  break;

  default:
        cout << " error " ;


  }
```

(3 points)

Question 5.

Explain what does the following recursive program returns,

```
int sum ( int N)
{
        int result;
        if (N= =1)
                result = 1;
        else
                result = N + sum (N-1);
        return  result ;
}
```

(2 points)

Question 6. Write a program to determine all integers less than 3,000,000 with the property that they equal the sum of factorials of their digits; 145=1!+4!+5! for example.

(5 points)

Programming Methodology I - COEN 243
Solution to Quiz #1

Solution 1:

The programmer clearly intended for the second output **"total is not equal to MAX"** to be printed if the first condition **(total == MAX)** is false, regardless of the second condition **(total < sum)**. That is, the **else** was intended to be matched with the first **if**. But the "**else** matching" rule causes it to be matched with the second condition, which means that the output **"total is not equal to MAX"** will be printed only when **total** is equal to **MAX** and **total** is not less than **sum.**

The "**else** matching" rule can be overridden with braces:

```
if ( total == MAX)  {
    if ( total < sum )
        cout << "total equals MAX and total is less than sum";
}
else
    cout << "total is not equal to MAX";
```

Now the **else** will be matched with the first **if**, the way the programmer had intended it to be.

Solution 2:

In this problem you were asked to find the value of the integer variables *total* **and** *num* after each of the two statements. It is also given that

total = 2,
and **num = 3** at the beginning of each statement

a) total = ++num;

Here num is pre-incremented by 1, then its value is used to calculate total.
The statement has the similar effect as the following statements:

```
num = num + 1;
total = num;
```

Therefore after executing the statements

num = 4
and total = 4

b) num = total++;

Here total is post-incremented by 1, that is, use the current value of total (which is 2), assign it to num, then increment total. The statement has the similar effect as the following statements:

```
num = total;            //total at the beginning of statement is given to be 2
total = total + 1;
```

Therefore after executing the statements

num = 2
and total = 2 + 1 = 3

Solution 3:

It is not intended here to write the whole program, only the part of the program that is given is to be rewritten using while loop.

In do-while loop **action** (as specified in its body) is performed first before evaluating the **condition,** whereas in while loop condition is first tested before performing the action. Therefore, one cycle of action may be needed before a while loop as described below:

```
M = 2;
S = 0;
I = 0;

// one cycle of do-while body
S = S + M;
M = 3 * M + I;
++I;
cout << "M= " << M << endl;

// same condition M < 100 to be tested
while (M < 100)
{
S = S + M;
M = 3 * M + I;
++I;
cout << "M= " << M << endl;
}
```

But since the condition M < 100 is true even in the first cycle of while loop, it may well be part of the while loop and the following code is good too:

```
M = 2;
S = 0;
I = 0;

// same condition M < 100 to be tested
while (M < 100)
{
S = S + M;
M = 3 * M + I;
++I;
cout << "M= " << M << endl;
}
```

Solution 4:

switch multiple-selection structure and the nested *if-else* structure has the same structures. You may rewrite the program using indentation or avoiding the deep indentation of the code to the right.

```
if ( (number == 1) || ( number == 2 ))
    cout << "digits";
else
    if ( number == 10 )
        cout << "tens";
```

```
else
    if ( number == 100 )
        cout << "hundreds";
    else
        if ( number == 1000 )
            cout << "thousands";
        else
            if ( number == 10000 )
                cout << "ten thousands";
            else
                cout << "error";
```

Without deep indentation

```
if ( (number == 1) || ( number == 2 ))
    cout << "digits";
else if ( number == 10 )
    cout << "tens";
else if ( number == 100 )
    cout << "hundreds";
else if ( number == 1000 )
    cout << "thousands";
else if ( number == 10000 )
    cout << "ten thousands";
else
    cout << "error";
```

Solution 5:

The recursive program returns the summation of integers from 1 to N. That is, the final return will result in

Result = N + (N-1) + (N-2) + ... + 2 + 1

Through the following recursive calls:

Result = N + sum(N-1)
= N + (N-1) + sum(N-2)
= N + (N-1) + (N-2) + sum(N-3)
:
:
= N + (N-1) + (N-2) + ... + (N-(N-2)) + sum(1)
= N + (N-1) + (N-2) + ... + 2 + 1

where sum(1) returns result =1 as the base case.

Solution 6:

```
// Integers with the property that they equal the sum of factorials of their digits
#include <iostream.h>

// Since it is clear from the problem that factorial needs to be calculated, you should
// provide a function for factorial
// data type of long is chosen for return-type since factorial of 9 would return a value
// greater than it would fit in 2 bytes.
// Function argument can be of int type since we need to calculate factorials of only 0 to 9
```

Department of Electrical and Computer Engineering
Concordia University

Programming Methodology I – COEN 243

Fall 2000 – Sections: U and W

Quiz #1 (Sunday Oct. 22, 2000)

Solve all questions
Time: 90 minutes

Total Marks = 15 points
Closed book.

Question 1. Suppose that the integer type was represented by 24-bit numbers. What ranges of numbers would be represented by **int** and **unsigned int**.

(1 point)

Question 2. What is the output of the following program segment? Be careful.

```
double p = 2.5, q, r;
q = 1 + 2 * p;
r = p - q;
cout << "2 * p + r";
```

(1.4 points)

Question 3. Who wrote this mess? The function definition below is full with errors. Identify all of them and explain each error.

```
int Bad Function(i,j);
{
  int i =1, j, sum;
  const int temporary;
  sum = i + j ;)
  temporary = i;
  j = temporary;
  i++ = temporary - sum
  cout >> j - i;
```

(3.6 points)

Question 4. Write the definition of a function

int Ordered(int p, int q, int r)

that returns 1 if and only if p, q, and r are in order, i.e., if p<=q<=r, and returns 0 if they are not.

(2 points)

Programming Methodology I - COEN 243

SOLUTION TO QUIZ 1

Question 1.

Range for int : -2^{23} to $(2^{23}-1)$

Range for unsigned int : 0 to $(2^{24}-1)$

Question 2.

The program segment will output,

2 * p + r

Question 3.

```
int Bad Function(i,j);
```

Error No. 1: Spaces not allowed in function names. They have to be a valid identifier of one word

Error No. 2: data types should be present in the parameter list

Error No. 3: semicolon not allowed. This is given as a function definition and not as a function prototype

```
{
  int i =1, j, sum;
```

Error No. 4: variables i and j are redeclared.

```
  const int temporary;
```

Error No. 5: temporary declared as constant should be initialized while declaration

```
  sum = i + j
```

Error No. 6: semicolon missing

```
  temporary = i;
```

Error No. 7: temporary is a constant and cannot be assigned a value

```
        j = temporary;
        i++ = temporary – sum
```
Error No. 8: lvalue cannot be an arithmetic expression
Error No. 9: semicolon missing

```
        cout >> j - i;
```
Error No. 10: cout operator wrong direction

Error No. 11: the function definition has a return type of int, the function should therefore return some value
Error No. 12: missing closing braces } for function definition

Question 4.

```
int Ordered( int p, int q, int r)
{
   if ( (p<=q) && (q<=r) )
        return 1;
   else
        return 0;
}
```

Question 5.

```
        int n = 0, sum = 0;

        do
        {
           sum += n;
           cout << "Entry? ";
           cin >> n;
        }
        while ( n != -1 );

        cout << "The sum is " << sum;
```

Do/while loop executes action at least once. Note that this is one of the solutions.
Here n is initialized to zero, such that the first pass in do loop gives correct value of sum.

Question 6.

```
int  function  reverse ( int number )
{
        int   r_value, temp, nofdigits= 0, r_number= 0;

        temp = number;


    // Determining the number of digits in the number

       while ( temp > 0 ) {

                temp /= 10;
                nofdigits++;
       }




 // Determining the reverse number

       while ( number > 0 ) {
                r_value= number % 10;  // Extract the right digit of the remaining number

                r_number += r_value*pow(10, nofdigits-1);  // constructing the reverse

                nofdigits--;

                number /= 10;     // Determine the remaining number

       }

       return r_number;

}
```

Department of Electrical and Computer Engineering
Concordia University

Programming Methodology I - COEN 243
Fall 1999 - Sections : U and W

Quiz #2 (Sunday Nov. 21, 1999)

Solve all six questions **Total Marks = 25 points**
Time: 120 minutes **Closed book.**

1. Determine the values stored in each of the variables (including the pointers) in the following program,

```
#include <iostream.h>
int main ( )
{   float x, y, *p, *q, *r;
    x =100.0;  q = &y;
    p = &x;   r = q;
   *r = x*2;
    cout << x << " " << y << endl;
 return 0;
}
```

(2.5 points)

2. The following program should exchange the contents of variables *i* and *j*, producing the following output, $i = 20$, $j = 10$
However, the program may have errors and missing statements, **explain the errors and correct the program.**

```
1 #include <iostream.h>
2
3 void swap ( int *, int *)
4 int main ( ) {
5    int i = 10, j = 20;
6    int *ip, jp;
7    ip = i;
8    jp = j;
9    swap (ip, jp)
10   cout  << " i = " << i << ", j = " << j << endl;
11    return 0;
12 }
13
14 void swap (int *p1, int *p2) {
15        *p1 = *p2;
16        *p2 = *p1; }
```

(3.5 points)

3. Given the following program, determine what do the output statements print.

```
#include <iostream.h>
 int main ( )
{  char color[] = "blue";
   char *colorPtr = color;

   cout << color [1] << colorPtr[3] << endl;
   cout << *(color+0) << *(colorPtr+2) << endl;
   colorPtr++;
```

```
    cout << *(colorPtr++) << colorPtr << endl;
    cout << *(color) << color << endl;
     return 0;
}
```

(4 points)

4. **Explain** what does the following program prints,

```
#include <iostream.h>

int mystery ( char *, const char *);

int main ( )
{ char string1[80], string2[80];


  cout << " Enter two strings : ";
   cin >> setw[30] >>string1 >> setw [20] >> string2;
   cout << mystery (string1, string2 ) <<string1 << endl;
   return 0;
}

int mystery ( char *s1, const char *s2)
{    int x = 1;
     while ( *s1 != '\0' ) {  ++s1;  ++x; }

      *s1 = ' ';

     for ( ; *s2 != '\0'; s2++ ) {
               s1++;
               ++x;
              *s1 = *s2;
     }
     return x;
 }
```

(5 points)

5. Given an array of integers, write a program that will print the ascending values in the array.
   Example : Given, int a [] = { 5, 3, 10, 13, 4, 40, 25, 6 }, the program should print 5, 10, 13, 40.
   (5 points)

6. Given a string array that consists of lower case letters, write a program to determine **the longest identical character patterns** in the string. The program should print the beginning subscripts of the patterns and their length. If there are more than two such identical patterns, then the program should print only the subscripts of any two of them. If there are no identical patterns, then the program should print " No matching patterns are found ".

   Example : aef**tskmn**hty**tskmn**ijaef
   The longest identical pattern in the above string is " **tskmn** ". The beginning subscripts of the two patterns are 3 and 11 respectively.
   (5 points)

Note : The quality of your programming will also be evaluated.

SOLUTION TO QUIZ 2

1. The variables have the following values,

   x = 100.0, y = 200

   p = address of x, q = r = address of y

2. The correct program is below,

```
#include <iostream.h>

void swap ( int *, int * );        // Semicolumn missing

int main ( )
{  int i = 10, j = 20;
   int *ip, *jp;                   // Asterisk missing from pointer jp
   ip = &i;                        // Address operator missing from i.
   jp = &j;                        // Address operator missing from j.
   swap ( ip, jp);                 // Semicolumn missing
   cout << "i=" << i << ",j = "<< j << endl;
    return 0;
}

void swap ( int *p1, int *p2 ) {

    int temp = *p1;     // This statement was missing.
    *p1 = *p2;
    *p2 = temp;         // temp instead of *p1 is needed.
}
```

3. It adds space to the end of string1 and then appends string2. It also determines the number of characters in the combined string.

   Example : String1 : Good String2 : Bye

   Output : 8Good Bye

4. color[1] = λ , colorPtr [3] = e

   *(color+0) = b , *(colorPtr +2) = u

*(colorPtr) = λ colorPtr = λue

*(color) = b color = blue

5.

```
#include <iostream.h>

int main ( )
{ int size = 8;
  int temp;
  int a[ ] = {5, 3, 10, 13, 4, 40, 25, 6};

  temp = a[0];

  cout << a[0] << endl;

  for ( int i = 1; i < size ; i++ ) {

        if ( a[i] > temp ) {

            cout < a[i] << endl;
             temp = a[i];
        }

  }
}
```

6.

```
#include <iostream.h>

int main ( )
{ char string [ ] = " aeftskmnhtytskmnijaef ";
  int s1, s2, len, n_len;
   len = 0;

  for ( int i = 1; string [i] != '\0'; i++) {

  // Compare the i' th character with all the characters before it.
```

```
        for ( int j = 0; j < i; j++ ) {
   // If i'th character is same as j'th character, this is the beginning of an identical pattern
   // We determine the length of this pattern.

                    n_len = 1;


          while (string [j+n_len] == string [i+n_len] ) {
                        n_len++;
                    }
 // Determine if we have found two patterns that have longer length.

                    if ( n_len > len ) {
                    s1 = j;
                    s2 = i;
                    len = n_len;
                    }
              }

    }

   if ( len > 0 ) {
     cout << " Beginning Subscripts of 1. and 2. Patterns " << s1 << << ',' <<s2<<endl;
     cout << " Length of the Pattern " << len<< endl;
     }
    else
       cout << " No matching patterns are found ";
2return 0;

}
```

```
long factorial( int );

int main()
{
  long num, number, sum_factorial;          // sum_factorial of type long to match with return-
                                            // type of function factorial
  int digit;                                // for each digit extracted from the integer
                                            // digit is of type int which matches with function call
                                            // for factorial
   for ( num = 1; num < 3000000; num++ )
   {
                sum_factorial = 0;          // initialize sum for every integer
                number = num;               // save original integer to be used later
                while (number > 0)
                {
                        digit = number % 10; // take the rightmost digit
                        sum_factorial += factorial( digit ); // find its factorial and add it to sum
                        number /= 10;       // divide integer by 10 so the next digit will be on the
                                            // right for the next pass through the loop.
                }
                if (sum_factorial == num)
                {
                   cout << num << endl;
                }
        }

  return 0;
}

// Definition of recursive function factorial
long factorial( int number )
{
  if ( number <= 1 )  // base case
    return 1;
  else              // recursive case
    return number * factorial( number - 1 );
}
```