ICOM4015 Otoño 2000 Exámen I 3/5/01

Nombre: _____ **Sección:** _____

¡Anota tu nombre y número de sección en <u>todas las hojas</u> del examen AHORA!

Tienes 2 horas para completar tres problemas. Lee cuidadosamente todo el examen antes de empezar a trabajar. Muestra todo el trabajo conducente a tu contestación. Podrás recibir crédito parcial por contestaciones parciales siempre y cuando muestres tu trabajo por escrito. Usa tu tiempo inteligentemente. Exito!

ICOM 4015 Staff

1	34
2	33
3	33
Total	100

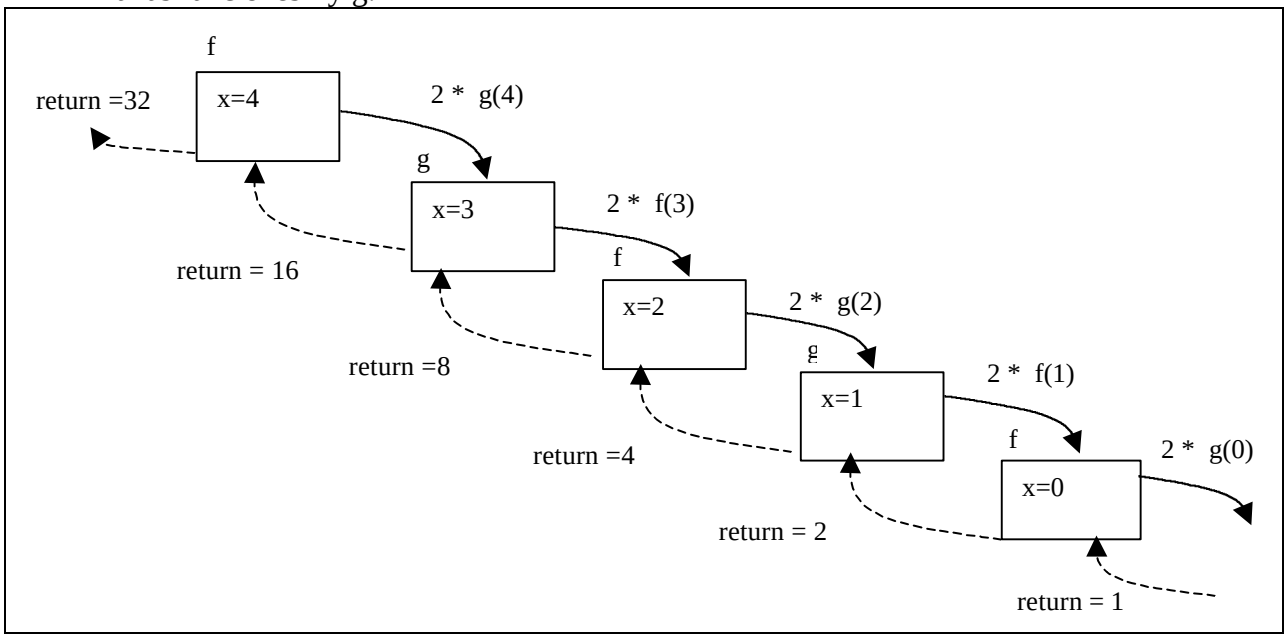
Nombre: _____ **Sección:** _____

Problema 1. Scope Rules / Recursion / Procedural Abstraction (34 puntos)

Contesta las preguntas a continuación basándote en el siguiente código:

```
#include <iostream>

int g(int  x);
int f(int &x);

int x = 5;

main()
{
  cout << "Result: " << f(x) << endl;
  cout << "x in main = " << x << endl;
}

int f(int &x){
  if (x == 0) return 1;
  x--;
  cout << "x in f = " << x << endl;
  return(2 *  g(x));
}

int g(int  x){
  if (x == 0) return 1;
  x--;
  cout << "x in g = " << x << endl;
  return(2 * f(x));
}
```

(a) (7 puntos) Demuestra la cadena de records de activación generados por las invocaciones a las funciones f y g:



¡OJO! Problema 1 continúa en la próxima página ...

Nombre: _____ **Sección:** _____

(b) (7 puntos) Cuál es el output generado por el programa?

```
x in f = 4
x in g = 3
x in f = 2
x in g = 1
x in f = 0
Result: 32
x in main  =  4
```

(c) (7 puntos) Cuál es el contrato de la función f(x)? Describe el contrato concisamente en términos de sus parámetros, su resultado y sus otros efectos secundarios, sin tomar en consideración los outputs (cout's) dentro de f(x) y g(x). En pocas palabras, que hace f(x)?

f(x) = 2^x
y disminuye x por un factor de 1

otra posible interpretacion
[x--,2^x] = f(x)

(d) (13 puntos) Escribe una función iterativa llamada *iterF* que satisfaga el mismo contrato de f. La función *iterF* debe tener el mismo prototipo que f.

```
int iterF (int & x) {

        int result = 1;
        for( int i = x ; i != 0 ; i--)
                result * =2;
        x--;
        return result;









}
```

Nombre: _____ **Sección:** _____

Problema 2. C++ Basico (33 puntos)

La función *factors* a continuación imprime los factores primos del numero que recibe como argumento.

```
int factors(long number) {
  int numFactors = 0;
  int quotient = number;
  // outer loop computes next quotient
  while (quotient > 1) {
    // inner loop finds next factor
    for (int factor = 2; factor <= quotient; ) {
      if (quotient % factor == 0) {
        cout << "Next factor is: " << factor << endl;
        numFactors++;
        quotient /= factor;
      }
      else {
        factor++;
      }
    }
  }
  return numFactors;
}
```

El output de la invocación de factors con argumento 48 es:

```
Next factor is: 2
Next factor is: 2
Next factor is: 2
Next factor is: 2
Next factor is: 3
48 has 5 factors
```

(a) (15 puntos) Escribe una nueva versión de *factors* llamada *polyFactors* que imprima el argumento representado como un producto de potencias de sus factores primos. La función puede asumir que el argumento siempre es un número positivo mayor que 2. Utiliza el símbolo '^' para representar la operación exponencial.

Ejemplos:

factors(60) debe imprimir "2^2 * 3 * 5"
factors(48) debe imprimir "2^4 * 3"
factors(16) debe imprimir "2^4"

¡OJO! Problema 2 continúa en la próxima página ...

ICOM4015 Otoño 2000 Exámen I 3/5/01

Nombre: _____ **Sección:** _____

```
int polyFactors(long number) {
  int numFactors = 0, countFactors = 0;
  int quotient = number;
  bool start1=false,start2=false;
  // outer loop computes next quotient
  while (quotient > 1) {
    // inner loop finds next factor
    for (int factor = 2; factor <= quotient;) {
      if (quotient % factor == 0) {
          if(start1 == false && start2 == false){
               cout<<factor;
               start1=start2=true;
          }
          if(start2 == false){
               cout<< "*" << factor;
               start2 = true;
          }
        numFactors++;
        countFactors++;
        quotient /= factor;
      }
      else {
        factor++;
        start2= false;
        if(countFactors >= 2)
          cout<< "^" << countFactors;
        countFactors=0
      }
    }
  }
  return numFactors;
}
```

Nombre: _____ **Sección:** _____

Problema 3. I/O Streams (33 puntos)

Escribe una función diamonds que reciba dos parámetros llamados *symbol* (tipo **char**) y *width* (tipo **int**). La función debe imprimir un diamante de ancho *width* relleno de símbolos como *symbol*. La función puede asumir que *width* siempre es un número impar mayor de 1.

Ejemplo:

diamods('*', 9) debe imprimir el siguiente diamante:

```
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

diamods('*', 3) debe imprimir el siguiente diamante:

```
  *
 ***
  *
```

```
void diamonds(char symbol, int width) {
     int space=width/2 +1;
     int i,j;
     for(i=1; i<=width;i+=2,space--){
          cout<<setw(space);
          cout<<symbol<<setw(0);
          for(j=i;j!=1;j--){
               cout<<symbol;
          }
          cout<<endl;
     }
     space=2;
     for(i=width-2;i>=1;i-=2,space++){
          cout<<setw(space);
          for(j=i;j!=0;j--){
               cout<<symbol;
          }
          cout<<endl;
     }
}
```