UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

INE018 MATEMÁTICA COMPUTACIONAL

Examen Parcial
2024-1

Indicaciones generales:

- Duración: 120 minutos.
- No está permitido el uso de ningún material o equipo electrónico adicional al indicado (no celulares, no tablets, no libros).
- **La presentación, la ortografía y la gramática de los trabajos influirán en la calificación.**

Puntaje total: 20 puntos.

Pregunta 1. (2 puntos)

¿Cuál es el nombre de la función que debe ser definida en todos los programas en C++? ¿Qué sentencia típicamente aparece al final de dicha función?

Pregunta 2. (2 puntos)

Indique los valores y tipos de las siguientes expresiones:

a. `2 + 3`

b. `19.0 / 5`

c. `19 % 5`

d. `6 + 5 / 4 - 3`

e. `10 + 9 * ((8 + 7) % 6) + 5 * 4 % 3 * 2 + 1`

f. `8 + 5 * 3 / 2`

g. `3 / 14 / 7 / (1.0 * 2) + 10 / 6`

h. `1 / 2 > 0 || 4 == 9 % 5 || 1 + 1 < 1 - 1`

Pregunta 3. (2 puntos)

¿Qué línea de control de bucle `for` usaría para contar de siete en siete comenzando desde cero hasta que el número tenga más de dos dígitos?

Pregunta 4. (2 puntos)

Las variables declaradas dentro de una función son llamadas *variables locales*. ¿Cuál es el significado de la palabra *local* en este contexto?

Pregunta 5. (2 puntos)

Justifique la veracidad o falsedad del siguiente enunciado:
Si ejecutamos las líneas

```
string linea;
cin >> linea;
```

el programa leerá la línea entera de datos del usuario y la guardará en la variable `linea`.

Pregunta 6. (2 puntos)

Dado el siguiente programa:

```
void Imprimir(int dos, int uno, int tres) {
    cout << tres << " es casi " << dos << " + " << uno << endl;
}

void Misterio(void) {
    int uno = 4;
    int dos = 3;
    int tres = 10;
    int n = 17;
    int cuatro = 3;
    Imprimir(uno, dos, tres);
    Imprimir(tres, cuatro, 5);
    Imprimir(2, dos * 2, n);
    Imprimir(n, tres * uno, cuatro);
    Imprimir(tres - cuatro, uno, dos);
}
```

Escriba la salida de cada una de las llamadas tal como aparecerían en la consola.

Pregunta 7. (2 puntos)

Para cada llamada a la función

```
int Misterio(int n) {
    if (n < 0) {
        n *= 3;
        return n;
    } else {
        n = n + 3;
    }
    if (n % 2 == 1) {
        n += n % 10;
```

```
    }
    return n;
}
```

escriba el valor retornado:

```
Misterio(-5);
Misterio(0);
Misterio(7);
Misterio(18);
Misterio(49);
```

Pregunta 8. (2 puntos)

Para cada llamada a la función

```
int Misterio(int i, int j) {
    int k = 0;
    while (i > j) {
        i = i - j;
        k = k + (i - 1);
    }
    return k;
}
```

escriba el valor retornado:

```
Misterio(2, 9);
Misterio(5, 1);
Misterio(38, 5);
Misterio(5, 9);
Misterio(40, 10);
```

Pregunta 9. (2 puntos)

Escriba un procedimiento llamado `ImprimirMultiplos` que acepte un entero n y un entero m como parámetros e imprima una línea reportando los m primeros múltiplos de n. Por ejemplo, las siguientes llamadas

```
ImprimirMultiplos(3, 5);
ImprimirMultiplos(7, 3);
```

deben producir la siguiente salida:

```
Los 5 primeros múltiplos de 3 son 3, 6, 9, 12 y 15.
Los 3 primeros múltiplos de 7 son 7, 14 y 21.
```

Note la separación entre los múltiplos y los signos de puntuación. Debe reproducir exactamente este formato. Asuma que el número de múltiplos solicitados es mayor que 1.

Pregunta 10. (2 puntos)

Escriba un procedimiento llamado `TresCaras` que repetitivamente lanza una moneda hasta obtener tres caras seguidas. Cada vez que la moneda es lanzada, se imprime lo obtenido (C para caras, S para sellos). Cuando se consiguen tres caras consecutivas, imprima un mensaje de felicitación. A continuación tiene una posible salida a una llamada a la función `TresCaras`:

```
S S S C S C C C
Tres caras consecutivas :)
```

Utilice la función `rand` de la biblioteca `<cstdlib>` para generar números pseudoaleatorios.

Observación: Si leemos la documentación de la función `rand`, encontramos que retorna un número pseudoaleatorio entre `0` y `RAND_MAX`. En la documentación de `RAND_MAX` encontramos que su valor depende de la implementación y solo nos garantiza que es al menos 32767. Desde C++11 existen mejores generadores de números pseudoaleatorios. En particular, la biblioteca `<random>` nos provee un Mersenne Twistter basado en el primo $2^{19937}-1$ llamado `mt19937`. Asimismo, en esta biblioteca encontramos distribuciones como `uniform_int_distribution` la cual nos da perfectamente números uniformemente distribuidos sin el sesgo del residuo. Yo suelo utilizar lo mencionado de la siguiente manera:

```cpp
int64_t seed = std::chrono::steady_clock::now().time_since_epoch().count();
std::mt19937_64 gen(seed);
long Random(long l, long r) {
    return std::uniform_int_distribution<long>(l, r)(gen);
}
```

Profesor del curso: Manuel Loaiza Vasquez.

Lima, 6 de junio de 2024.