?? marks **CS 101 Endsem** **2:30-5:30, 26/11/12**

On the top right hand corner of your answerbook, please write the timings of your lab batch.

If a question has several parts, you may answer the later parts even if you did not answer the preceding ones.

Problem 1(a) [10 marks] Given below is an infinite product that approaches $2/\pi$. Write a function which returns the products of the first n terms. The function should take n as an argument.

$$\frac{\sqrt{2}}{2} \cdot \frac{\sqrt{2+\sqrt{2}}}{2} \cdot \frac{\sqrt{2+\sqrt{2+\sqrt{2}}}}{2} \cdots$$

```
double twobypi(int n){            // 1 mark, declaration
 double result=1, num=sqrt(2);   // 1 marks for allowing n=0
 for(int i=0; i<n; i++){
    result *= num/2;           // 3 marks terms correctly calculated
    num = sqrt(2 + num);       // 4 marks: correct final answer
 }
 return result;               // 1 mark
}
```

(b)[8 marks] Give code to create a vector of vector of doubles, representing a 5×5 identity matrix.

```
 vector<vector <double> > m(5, vector<double>(5,0));
 for(int i=0; i<5; i++) m[i][i] = 1;
// 3 marks: correct type of the datum, i.e. vector<vector<double> >
// 3 marks: allocate memory to get the correct shape
// 2 marks: assign values correctly.
```

(c)[7 marks] What will you see when the following program waits for a click? Explain why.

```
int main(){
 initCanvas("Fun",500,500);
 Line d(0,0,500,500);
 for(int i=1; i<5; i++){
    Circle c(250,250,20*i);  c.imprint();
    Line l(250,250,300,20*i);
 }
 getClick();
}
```

You will see 4 circles because they are imprinted.

You will see a diagonal line because the associated variable is still in the activation frame.

You will not see 4 lines because they will have been created but the variables will have been deallocated.

4 marks for the correct answer: diagonal + circles and no other lines. No partial credit in this.

3 marks for the correct explanation, one for each above.

Problem 2(a)[15 marks] Define a class for representing a triangle (not for drawing on the screen). It should have (non public) data members to hold the coordinates of the vertices. It should have a default constructor that sets all coordinates to 0. It should also have a member function `initialize` that takes values of the coordinates as arguments and sets the data members accordingly. It should also have a member function which returns the area of the triangle, to be calculated using the following expression:

$$|(x_1y_3 + y_1x_2 + x_3y_2 - x_2y_3 - y_2x_1 - y_1x_3)/2|$$

where (x_1, y_1), (x_2, y_2) and (x_3, y_3) are the coordinates of the vertices.

(b)[5 marks] Inherit from the class you defined above and add a member function perimeter which returns the perimeter of the triangle.

```
class triangle{             // 1 mark
protected:                  // 2 marks, marks deducted for private access modifier
 double x1,x2,x3,y1,y2,y3; // 2 marks, 1 mark deducted for declaring it as int
public:                     // 2 marks
 triangle(){x1=x2=x3=y1=y2=y3=0;} // 3 marks deducted 1 mark for return type
 void reset(double a1, double b1, // 2 marks
            double a2, double b2, double a3, double b3){
    x1 = a1; x2 = a2; x3 = a3;
    y1 = b1; y2 = b2; y3 = b3;
 }
 double area(){                   // 3 marks, deducted 1 mark for not using abs()
// deducted 1 marks if they have declared class variables as int and not using typecasting
// or having int  as return type
    return abs(x1*y3+y1*x2+x3*y2 - x2*y3 - y2*x1 - y1*x3)/2;
 }
};

class nTriangle : public triangle{  // 2 marks
public:                              // 1 mark, reduced 1 marks for not writting public
 double perimeter(){                // 2 marks
    return sqrt(pow(x1-x2,2)+pow(y1-y2,2)) +
     sqrt(pow(x2-x3,2)+pow(y2-y3,2)) + sqrt(pow(x3-x1,2)+pow(y3-y1,2));
 }
};
```

Problem 3[20 marks] Write a program which takes a sequence of numbers from the keyboard and prints "Eureka!" whenever the last 50 numbers received till then are odd, and the 50 preceding them are even. In other words, for all i, on reading the ith number x_i , "Eureka!" should be printed if $x_{i-99}, \ldots, x_{i-50}$ are even, and $x_{i-49}, \ldots, x_i$ are odd. Do not use arrays, nor more than 3,4 variables. Clearly write down the loop invariants. Briefly explain why the loop invariants hold.

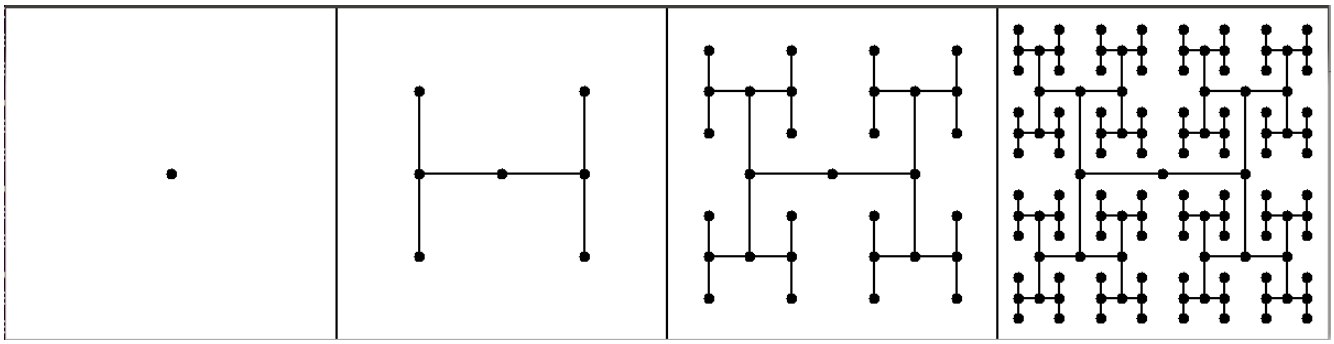
```
int main(){
 int v;
```

```
 int found=0;  // invariant: found will denote the number of matches
               // so far.     4 marks
 while(cin >> v){
    if(found < 50){         // 4 marks
     if(v%2 == 0) found++;
     else found = 0;
    }
    else if(found == 50){   // 4 marks
     if(v%2 == 0){}
     else found++;
    }
    else if(found < 99){    // 4 marks
     if(v%2 == 0) found = 1;
     else found++;
    }
    else{ // found = 99     // 4 marks
     if(v%2 == 0) found = 1;
     else{
       cout << "Eureka!\n";
       found = 0;
     }
    }
 }
}
/*
0 marks if arrays, vectors etc are used.

4 marks for stating the invariant clearly.  This may be stated
differently, e.g. ‘‘phase = 1 and count = x denotes that only x evens
have been seen, while phase=2 and count=x denotes x odds have been
seen after 50 evens have been seen’’

There are 4 cases depending upon the state.  If the code for a state
is completely correct, then give 4 marks.  No partial credit, i.e. do
not give any marks if in some case correct updating happens on odd
input but no even.
The student might not have written the answer in the above, manner,
but you should see what the program does for each of the cases above.
*/
```

Problem 4[15 marks] Binary trees are typically drawn with the root at the top and branches growing downwards. However, it is also useful to draw trees differently. For example, trees with even height can be drawn as shown below, for heights 0, 2, 4 and 6. Such layouts are useful in circuit design. Write a recursive function to generate such a layout, using the declaration as follows:

```
void Htree(double cx, double cy, int hby2, double SL);
```

The root should be at coordinates `(cx,cy)`, and `hby2` denotes half the tree height, and `SL` the length of the side of the (square) layout.



```
void Htree(double cx, double cy, int hby2, double SL){
 Circle c(cx,cy,5);     // base case
 c.setFill();
 c.imprint();
 if(hby2 == 0) return;  // base case ends
 {Circle c(cx-SL/4,cy,5);
 c.setFill();
 c.imprint();}
 {Circle c(cx+SL/4,cy,5);
 c.setFill();
 c.imprint();}
 Line(cx-SL/4,cy,cx+SL/4,cy).imprint();
 Line(cx-SL/4,cy-SL/4,cx-SL/4,cy+SL/4).imprint();
 Line(cx+SL/4,cy-SL/4,cx+SL/4,cy+SL/4).imprint();
 Htree(cx-SL/4,cy-SL/4, hby2-1, SL/2);
 Htree(cx-SL/4,cy+SL/4, hby2-1, SL/2);
 Htree(cx+SL/4,cy-SL/4, hby2-1, SL/2);
 Htree(cx+SL/4,cy+SL/4, hby2-1, SL/2);
}
/*
base case: do nothing: 2 marks, put a circle 3 marks

4 calls to Htree.  3 marks
calls centered at roughly proper places  1 mark
hby2 reduces by 1                        1 mark
side length reduces, 1 mark, by 1/2      2 mark

H is drawn correctly.                    3 marks, 1 per line.

circles at odd levels put down correctly 2 marks,

Reduce 1 mark if H is not imprinted, 1 more if circles are not
imprinted.  (reduce even if one line or or one circle is not imprinted).
```

```
*/
```

Problem 5:[25 marks] The inputs to this problem are the positive integers $a_0, \ldots, a_9$ and $b_0, \ldots, b_9$. The goal is to calculate

$$r = \frac{a_0 \times a_1 \times \cdots \times a_9}{b_0 \times b_1 \times \cdots \times b_9}$$

It is known that all a_i, b_i and r are integers smaller than 2^{32}. It is possible, however, that the numerator or denominator of the above fraction are larger than 2^{32}.

Idea 1: do not multiply but cancel out common factors.

Idea 2: Since the result is an integer, each element of the denominator will vanish.

Idea 3: When we finally multiply the numbers, no number will become too large, because their product is known to be small.

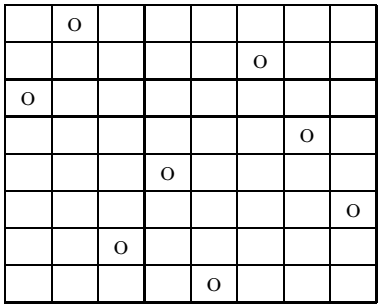
Give code that determines r assuming you have already read the integers into `unsigned int` arrays `a[10], b[10]` respectively. Clearly explain your code, and mention why the numbers you encounter in your calculation will always remain smaller than 2^{32}. If you need to use any function that has been discussed in class (e.g. `gcd`), state what it does and use it without defining it. Hint: how do you simplify such fractions using paper and pencil?

```
for(int i=0; i<10; i++)
   for(int j=0; j<10; j++){
    int g = gcd(b[i], a[j]);
    b[i] /= g;
    a[j] /= g;
   }

int r = 1;
for(int i=0; i<10; i++) r *= a[i];

// Idea of cancelling out factors is present: 5 marks
// for each b[i] removing all common factors from all a[j]: 5 marks
// using gcd rather than searching factors linearly: 5 marks
// final product computation                              : 5 marks
// Explaining final product does not overflow:
// because r is given to be small,
// and b[i] have become 1                          : 5 marks, no partial
// credit for this.  However, it is OK if the fact that b[i] becomes 1
// is mentioned somewhere earlier.
//
// 0 marks for throwing away remainders, or doing real division.
```

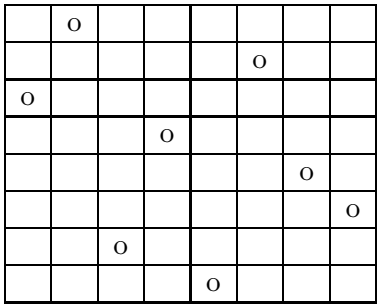
Problem 6: Consider an $n \times n$ array of squares, with rows and columns numbered 0 through $n-1$. We wish to place n coins, one in some square in each column, such that (a) there is at most

one coin in each row, and (b) if squares s, t contain coins, then s is not exactly to the Northeast or to the Northwest of t. A coin configuration is *good* if it satisfies the above requirements. Two examples are given below, for $n = 8$. The second example is bad because the coin in column 5 is exactly to the Northeast of the one in column 3, and also because the coin in column 6 is exactly to the Northwest of the one in column 7.

Good:



Bad:



(a)[15 marks] Write a function `good` having the following declaration:

```
bool good(int rowpos[], int n);
```

In this `rowpos` is expected to be an array of length `n` and `rowpos[i]` is expected to contain the row position of the coin in the `i`th column. The function is expected to return `true` if the coin configration indicated in the array is good, and `false` otherwise. Note that `false` must be returned even if the array contains an invalid configuration, i.e. the values are not in the range 0 through `n - 1`.

```
bool good(int rowpos[], int ncols){
 for(int i=0; i<ncols; i++)
    if(rowpos[i] < 0 ||
      rowpos[i] >= ncols) return false;  // invalid configuration.

 for(int i=1; i<ncols; i++){
    for(int j=0; j<i; j++){
     if(rowpos[i] == rowpos[j] ||           // same row
        abs(rowpos[i] - rowpos[j]) == i-j){ // NE or NW
       return false;
     }
    }
 }
 return true;
}
// 3 marks for returning false if invalid.
// 3 marks for returning false if same row.
// 3 marks for returning false if NE
// 3 marks for returning false if NW
// 3 marks for returning true otherwise (if above cases are written other wise 0) .
```

(b)[10 marks] Suppose you are given a function `explode` having the following declaration.

```
void explode(int rowpos[], int n, unsigned int configuration);
```

In this, `rowpos` is expected to be an array of length `n`. The parameter `configuration` is supposed to contain a number whose decimal digits give the row positions. The function sets `rowpos[i]` to the `i`th least significant digit of `configuration`. Thus, if `configuration` has the value 53174602, and `n` is 8, then the function will return 2, 0, 6, 4, 7, 1, 3, 5 in `rowpos[0]` through `rowpos[7]`.

Complete the following code fragment which uses `explode` to print all possible good configurations of an 8×8 board. Explain your answer. Will this idea also work for a 10×10 board?

```
int rowpos[8];
for(int i= _________; i < _________ ; i++){
 explode(rowpos, 8, i);
 if(good(rowpos, 8)) cout << i << endl;
}
```

i = 00000000 : 2 marks

j = 77777777 or 99999999 : 2 marks

1 mark for explaining these values.

Better value of i : e.g. 01234567 : 1 mark only if there is explanation

Better value of j : e.g. 76543210 : 1 mark only if there is explanation

will not work for 10x10 : 1 mark

because it will not fit in int: 2 marks

Problem 7:(a)[15 marks] In the `String` class we implemented in the lectures, the value of a variable was stored as a distinct copy in the heap memory. In this problem we will see how to use a single copy if two variables have the same value. The idea is to have a struct `sharedVal` which contains a pointer to the value, and a count of how many variables have this value. When we make assignments such as `x=y`, the number of variables using the value that `y` has increases, and the number using the value that `x` has decreases. When the number of variables using a value drops to 0 we return memory back to the heap. Given below is a sketch of the `String` class using this idea. Fill in the code for the functions `operator=` and `print`.

```
struct sharedVal{ int count;  char *data;  };

class String{
 sharedVal *r;
public:
 String(const char* rhs){
    r = new sharedVal;
    r->data = new char[length(rhs)+1];  // length : returns string length
    strcpy(r->data,rhs);                // strcpy(u,v) : copies v to u
    r->count = 1;
 }
```

```
 void operator=(const String &rhs){  ___give code___ }
 void print(){ ____give code.  Should print the value_________ }
}


//Solution-----------------------------------------------------------------

 void operator=(const String &rhs){
    r->count--;                                        // 3 marks
    if(r->count == 0){delete[] r->data; delete r;} // 3 marks
    r = rhs.r;                                         // 3 marks
    r->count++;                                        // 3 marks
 }                 // 2 marks extra: if(&rhs == this) return
 void print(){
    cout << r->data;   // 3 marks (with or without endl)
 }
```

(b)[5 marks] The code below uses the `string` class from the standard library. What will it print?

```
string x = "ab";
for(int i=0; i<5; i++) x = x + x;
cout << x << endl;
```

Will print out "ab" followed by itself for a total of 32 times. No partial credit.