Master M1 ENG/ENV/TECH Computer Programming Class 2017/18

Exam 2 hours, open book.

Answer to this exam must be given on the subject itself (all ___ must be filled) and on the exam sheets of paper. **When collected, if the student forgets to give back the subject (even if they do not answer any question on it), they will get a zero for the exam.** Answers that must be filled on the subject will be only considered if written on the subject not on the exam paper. This exam is graded out of 20.

Last Name: _____

First Name: _____

Room rank ID: _____

Exercise 1 (1 point out of 20)

++ increment operator:

a) Describe the two general ways of using this operator.

b) Describe its use with pointers.

Don't write pages and pages for this question.

Exercise 2 (1pt)

An algorithm has to test a huge number of times if some computed integer is or is not in a collection of integers created before. In C++ (2011 norm) what is the adapted STL container that could be used to store this collection so that those tests are intrinsically optimized by the container. The container is supposed to store the collection and nothing more.

Give the name of this container here: _____

Exercise 3 (1pt)

a) What g++ option is needed to compile a program that you intend to use with gdb ?

Just give the option, nothing more, here: _____

b) What is gdb and what is its purpose ?

c) Is a private member of an instance of a class accessible within gdb ?

Just answer yes or no here: _____

Exercise 4 (2pts)

a) How do you declare a template parameter ?

b) How many template parameters could you declare for a function or a class ?

c) How do you use a template parameter ?

d) What is “automatic parameter type deduction” ?

Exercise 5 (5pts)

The polygon hierarchy from lecture 5 is recapped in file polygon.h below :

```
1       class Polygon
2       {
3         protected:
4           double width, height;
5         public:
6           void set_values(double a, double  b);
7           virtual double area(void) =0;
8       };
9       class Rectangle: public Polygon
10      {
11        public:
12          double area ();
13      };
14      class Triangle: public Polygon
15      {
16        public:
17          double area ();
18      };
```

Implementation file polygon.cc is given here:

```
1       #include “polygon.h”
2       void Polygon::set_values(double a,double b)
3       {
4          width=a; height=b;
5       }
6       double Rectangle::area()
7       {
8          return(width * height);
9       }
10      double Triangle::area()
11      {
12         return(width * height/2);
13      }
```

This hierarchy is used in the following program :

```
1       #include “polygon.h”
2       int main()
3       {
4          Polygon *poly[3];
5          Rectangle r1;
6          Rectangle r2;
7          Triangle t1;
8          poly[0]=&r1;
9          poly[1]=&r2;
10         poly[2]=&t1;
11
12         for (int i=0;i<3;i++)
13         {
14             poly[i]->set_values(1.*(i+1),2.*(i+1));
15         }
16
17         for (int i=0;i<3;i++)
18         {
19             cout<<poly[i]->area()<<endl;
20         }
21
22         return 0;
23      }
```

a) Give the output of this program here:

____
____
____

b) In the main is the following possible:

```
      Polygon one_poly;
```

Just answer yes or no here: _____
Comment your answer (i.e. why it is yes or why it is no) here:
_____

c) We would like to introduce **fake** arithmetic in this hierarchy so that the following lines may be possible in the main (before the return in line 21 for example):

```
Rectangle r3 = t1 + t1;
Rectangle r4 = r1 + r2;
```

From an arithmetic point of view we are going to say that the addition of 2 triangles gives a rectangle whose height and width are respectively the sum of the height of both triangles and the sum of the width of both triangles.

The addition of 2 rectangles gives a rectangle whose height and width are respectively the sum of the height of both rectangles and the sum of the width of both rectangles.

Implement methods in appropriate files (just write new lines and the original line number where you do this addition) so that lines above, added to main function, compile and give expected results.

Exercise 6 (5 pts)

In mathematics, particularly linear algebra and numerical analysis, the Gram–Schmidt process is a method for orthonormalising a set of vectors in an inner product space, most commonly the Euclidean space $\mathbb{R}^n$ equipped with the standard inner product. The Gram–Schmidt process takes a finite, linearly independent set of vectors S = { v_1 , ..., v_k } for k ≤ n and generates an orthogonal set of vectors S′ = { e_1 , ..., e_k } that spans the same k-dimensional subspace of $\mathbb{R}^n$ as S.

The algorithm to obtain S' is the following :

- Let's define the projection operator by $project_u(v) = \frac{u \cdot v}{u \cdot u} u$ where $u \cdot v = u^t v$ is the dot (or inner) product in $\mathbb{R}^n$.
- The first vector of S' is $e_1 = \frac{u_1}{\|u_1\|}$ where $u_1 = v_1$ and $\|u\| = \sqrt{u \cdot u}$ is u Euclidean norm.
- The second vector of S' is $e_2 = \frac{u_2}{\|u_2\|}$ where $u_2 = v_2 - project_{u_1}(v_2)$
- The j[th] (j>1) vector of S' is computed as $e_j = \frac{u_j}{\|u_j\|}$ where $u_j = v_j - \sum_{i=1}^{i=j-1} project_{u_i}(v_j)$

Implement this algorithm just as if it was written in a main.cc file that compiles with g++, following requirements below :

- All vectors or set of vectors will be stored in double mono-dimensional array.
- Your program will use 3 generic functions (no more) that you will implement before your main function:
  - “dot” returning the dot product (double) of the two vectors given as argument by double pointers and a unique integer describing the dimension of these vectors.
  - “norm” returning the Euclidean norm (double) of a vector given as argument by a pointer and an integer describing its dimension.
  - “proj” returning nothing and computing $project_u(v)$ from the two vectors u and v given as argument by pointers. Computational result is stored in a third vector given as argument by a double pointer. A unique integer, passed as argument, describes the dimension of these 3 vectors.
- To simplify exercise n will be 4 and k will be 3. Normally by changing space from $\mathbb{R}^4$ to $\mathbb{R}^{10}$ for example, in your main, function “dot”, ”norm” and “proj” should remain unchanged.
- Your main function first asks the user to input the 3 $\mathbb{R}^4$ vectors v_i of S and stores them in a double mono-dimensional array V of size 3x4=12 (choose the correct storage to be able to use function above).
- Your program uses then “dot”, ”norm” and “proj” functions to compute S' that you will store in a double array E of size 12.
- Then your program will output on the terminal the content of E (i.e. the 3 $\mathbb{R}^4$ vectors e_i).

Reuse as much as possible “dot”,”norm” and “proj” where you can. Do simple input/output.

Exercise 7 (5pts)

What is the output when the following valid code is executed ?

```
1       #include <iostream>
2       using namespace std;
3       int main()
4       {
5           int i,j,a = 7,b = 8, n=5;
6           int A[5] = {3,2,3,6,9};
7           i = A[0];
8           j = A[1];
9           int *p = &A[i];
10          int *q = &A[j];
11          for (int k = 0; A[i] < b; ++k)
12          {
13              switch (( *p )%3)
14              {
15                  case 0 :
16                   {
17                       *q=(k+1)*a;
18                       i=(j++)%n;
19                       j%=n;
20                       break;
21                   }
22                  case 1 :
23                   {
24                       i = ( i+1 )%n;
25                       j = ( j+1 )%n;
26                   }
27                  case 2 :
28                   {
29                       q = p;
30                       i=j;
31                       break;
32                   }
33              }
34              p=&A[i];
35              q=&A[j];
36              cout<<"k = "<<k<<" i = "<<i<<" j = "<<j<<" A[i] = "<<A[i]<<endl;
37          }
38          cout<<A[0]<<" "<<A[1]<<endl;
39          cout<<A[2]<<" "<<A[3]<<endl;
40          cout<<A[4]<<endl;
41      }
```

A total of 13 integers are expected. Fill the form bellow:

k = ___ i = ___ j = ___ A[i] = ___

k = ___ i = ___ j = ___ A[i] = ___

___ ___

___ ___

___