Exam 2 hours, open book.

Exercise 1 (3pt)

"if","switch","for","do while"and "while" are the main 5 conditional structures in C++.

a) Give for each if they are testing or looping structure

b) Give for each how they work (describe their algorithmic components)

Exercise 2 (3pt)

In C++ a function is treating a task by running a set of instructions for a set of information given to it. It returns then a result. Don't forget template programming to answer this exercise (in particular for a and b).

a) What are the 4 ways to pass information (variable) to a function ?

b) What are the 2 ways to set types of those passed information when we declare the function ?

c) How results of task computation are given back to the user of the function ?

Exercise 3 (5pt)

What is the output when the following valid code fragment is executed (less than 5 lines are expected) ?

```
1       int a[] = {11,7,8,1};
2       int k = 1;
3       int *p = &a[k];
4       int n = 2;
5       do
6       {
7          switch (( *p )%3)
8          {
9             case 0 :
10            {
11               if (p != &a[3]) ++p;
12               else p=&a[0];
13               ++(*p);
14            }
15            case 1 :
16            {
17               k += n;
18               n = ( k%4 ) ? ( k/4 ) : ( k+2 )/4;
19               p = &a[k-n*4];
20               p[0] += 4;
21               cout<<"1 n = "<<n<<" k = "<<k<<" a = "<<a[0]<<" "<<a[1];
22               cout<<" "<<a[2]<<" "<<a[3]<<endl;
23               break;
24            }
25            case 2 :
26            {
27               k -= n;
28               n += ( k + 6 )-k/2;
29               p = &a[n%4];
30               p[0] += 1;
31               cout<<"2 n = "<<n<<" k = "<<k<<" a = "<<a[0]<<" "<<a[1];
32               cout <<" "<<a[2]<<" "<<a[3]<<endl;
33               break;
34             }
35         }
36      } while ( ( a[0]+a[1]+a[2]+a[3] ) < 42);
```

Exercise 4 (4pt)

A printer library is given by following header printer.h :

```
1       #include <iostream>
2       using namespace std;
3       class printer
4       {
5           public:
6               virtual void print(int i) = 0;
7               bool status(void);
8           private:
9               int resource;
10              int rmx;
11      };
12      class laser : public printer
13      {
14          public:
15              laser(void);
16              virtual void print(int i);
17      };
18      class inkjet : public printer
19      {
20          public:
21              inkjet(void);
22              virtual void print(int i);
23      };
24      class colorlaser : public laser
25      {
26          public:
27              colorlaser(void);
28              void print(int i);
29          private:
30              int resourcec;
31              int rmxc;
32      };
```

and implementation file printer.cc :

```
1       #include “printer.h”
2       inkjet::inkjet()
3       {
4           rmx = 20;
5           resource = rmx;
6       }
7       void inkjet::print(int i)
8       {
9           cout<<"printing "<<i<<" pages (inkjet)"<<endl;
10          resource -= i;
11
12      }
13      laser::laser()
14      {
15          rmx = 10;
16          resource = rmx;
17      }
```

```
18      void laser::print(int i)
19      {
20          cout<<"printing "<<i<<" pages (laser)"<<endl;
21          resource -= i;
22
23      }
24      colorlaser::colorlaser()
25      {
26          rmxc = 5;
27          resourcec = rmxc;
28      }
29      void colorlaser::print(int i)
30      {
31          cout<<"printing "<<i<<" pages (color laser)"<<endl;
32          resource -= i;
33          resourcec -= i;
34
35      }
36      bool printer::status()
37      {
38          cout<<"Resource level "<<100.*resource/rmx<<"%"<<endl;
39          if (resource < 1) return false;
40          else return true;
41      }
```

This library is used in the following program :

```
1       #include “printer.h”
2       int main()
3       {
4           printer * pool[3];
5           printer A;
6           laser B;
7           colorlaser C;
8           inkjet D;
9           pool[0] = &B;
10          pool[1] = &C;
11          pool[2] = &D;
12          for (int k = 0; k < 3; k++)
13          {
14              printer *p = pool[k];
15              int i = 5+(((k+1)%3)?0:5);
16              while (p->status())
17              {
18                  p->print(i);
19              }
20          }
21          return 0;
22      }
```

In this library the "resource" data member corresponds to consumable that you find in most printers (ink cartridge, powder bloc, ...). The “rmx” data member corresponds to a consumable in its initial state (brand new/never used). When pages are printed, consumable are used and “resource” is decreased. “laser” and “inkjet” classes represent black and white printer concept. “colorlaser” class represents the concept of a color laser printer.

a) This set of files does not compile. Correct the 2 errors and explain your solution. You must not modify more than 2 lines and should not modify the algorithm.

b) Now that you have an executable, print the outputs when you run it.

c) From a logical point of view you see in this example that something is not correctly taken into account. In this implementation we have a “resourcec” (with associated “rmxc”) data member which is not modified. This extra data member is here to add consumable for color. Add something to the class hierarchy so that color laser stops printing (looping) when one of its consumable is less than 1.

Exercise 5 (5 pts)

Composite Simpson's rule is a formula that approximates the integral of any function f over an interval $[a,b]$ split up into n subintervals, with n an even number :

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\sum_{j=1}^{n/2}\left[f(x_{2j-2}) + 4f(x_{2j-1}) + f(x_{2j})\right]$$

where $x_k = a + k \times h$ with $k \in [0,n]$ and $h = \frac{b-a}{n}$

We want to have a function template that does this computation for any $[a,b]$ interval, for any n, for any f and for any x type.

You will consider the following declaration :

```
1       template <typename T>
2       T simpson(const T & a, const T & b, int n, T (*func)(const T &x) );
3       #include “simpson_imp.h”
```

stored in header simpson.h

a) Give the implementation of this function (expected to be stored in simpson_imp.h). Here “func” is a pointer argument that represents any function f having one constant T type argument passed by reference and returning a T instance. Your implementation must check that n is even. If it is not the case it outputs a warning and replaces n by $n-1$ (or 2 if $n \leqslant 1$).
For example the following program should work with your implementation where T is of type double in this case:

```
1       #include<iostream>
2       #include <cmath>
3       #include “simpson.h”
4       using namespace std;
5       double mySimpleFunction(const double &x)
6       {
7           return 2*sin(x/2);
8       }
9       int main ()
10      {
11          double a = 0.;
12          double b = 1.;
13          cout<<simpson(a,b,10,mySimpleFunction)<<endl;
14          return 0;
15      }
```

Note that in the scope of simpson function “func” is to be used like any other function call. “func(z)” will return a T value corresponding to computation by function “func” of “z” argument.

b) Now imagine that we want to do some computation in complex arithmetic. Create the class “Complex” that works with the following program :

```
1       #include<iostream>
2       #include <cmath>
3       #include “simpson.h”
4       #include “complex.h”
5       using namespace std;
6       Complex mySimpleCompleFunction(const Complex &x)
7       {
8           return x+x;
9       }
10      int main ()
11      {
12          Complex  a(0.,1);
13          Complex b( 1.,0.);
14          cout<<simpson(a,b,10,mySimpleComplexFunction)<<endl;
15          return 0;
16      }
```