Question 1 (30%, each of the 3 sub-questions below is worth 10%).

(a) What is the output of the following code? Justify your answer.

```
#include<iostream>
using namespace std;
class A {
public:
   void print_class_name()  { cout <<" I am A"<<endl; }
};

class B : public A {
public:
   void print_class_name() { cout <<"I am B"<<endl; }
};

class C: public B { };

int main()
{
  C c;
  c.print_class_name();
  return 0;
}
```

(b) The code below will not compile, explain why.

```
#include<iostream>
using namespace std;

class I_AM_Base {};

class I_AM_Derived: public I_AM_Base {};

int main()
{
    I_AM_Base *bp = new I_AM_Derived;
    I_AM_Derived *dp = new I_AM_Base;
}
```

(c) The code below also will not compile, explain why and correct the mistake.

```
#include<iostream>
using namespace std;

class I_AM_Base
{
public :
    int x, y;
public:
   I_AM_Base(int i, int j){ x = i; y = j; }
};

class I_AM_Derived : public I_AM_Base
{
public:
   I_AM_Derived(int i, int j){x=i;x=j}
   void print() {cout << x+y <<endl; }
};

int main(void)
{
   I_AM_Derived q(1, 1);
   q.print();
   return 0;
}
```

Question 2 (40%, Part (a) 5%,Part (b) 5% Part (c) 15% Part (c) 15%).

(a) What are C++ function and class templates?

(b) Briefly explain the advantages and disadvantages of using templates in C++.

(c) What steps can be used to mitigate the disadvantages of templates.

(d) Write the code for the `Swap` function used in the `main()` below and provide the output of the program below. The Swap function takes two arguments as input and swaps their values. You function should work for both double and int data types (including other C++ data types).

```
#include <iostream>
using namespace std;

int main()
{
        int i1 = 1, i2 = -1;
        double d1 = -1.2, d2 = -2.5;


        cout << "Before calling Swap\n";
        cout << "i1 = " << i1 << ", i2 = " << i2<<endl;
        cout << "d1 = " << d1 << ", d2 = " << d2<<endl;;

        Swap(i1, i2);
        Swap(d1, d2);

    cout << "After calling Swap"<<endl;
        cout << "i1 = " << i1 << ", i2 = " << i2<<endl;
        cout << "d1 = " << d1 << ", d2 = " << d2<<endl;;

        return 0;
}
```

Question 3 (30%, each of the 3 subquestions below is worth 5%,20% and 5%).

(a) Derive expressions for the upper and lower boundary conditions of the Black-Scholes equation for a European put option.

(b) Consider the general parabolic partial differential equation below,

$$\begin{aligned}
&\frac{\partial v(t,x)}{\partial t} = a(t,x)\frac{\partial^2 v(t,x)}{\partial x^2} + b(t,x)\frac{\partial v(t,x)}{\partial x} + c(t,x)v(t,x) + d(t,x),\\
&V(T,X) = f(x),\\
&V(t,x_l) = f_l(x),\\
&V(t,x_u) = f_u(x),
\end{aligned}$$

where we look for a solution in $[0,T] \times [x_l, x_u]$ with $x_l < x_u$. Use forward differencing for the time derivative, centered differencing for the first order spatial derivative and a centered second difference for the diffusion term to derive the explicit finite difference method.

(c) When should the finite difference method be preferred over Monte-Carlo methods?

Answer 1 (30%, each of the 3 sub-questions below is worth 10%).

(a) The correct output is,

```
I am B
```

The print_class_name() function is not present in class C. So it is looked up in the inheritance hierarchy. print_class_name() is present in both classes A and B. In multilevel inheritance, then function is linearly searched up in the inheritance hierarchy until a matching function is found, in this case the first match is B.

(b) A base class pointer/reference can point/refer to a derived class object, but the other way is not possible.

(c) The base class members are not initialized. We should call the base class constructor in order to initialize base class members: `I_AM_Derived(int i, int j): I_AM_Base(i, j)`

Answer 2 (40%, Part (a) 5%,Part (b) 5% Part (c) 15% Part (c) 15%).

(a) A function template works in a similar way to normal functions except that single function template can work with different data types at once but, a single normal function can only work with one set of data types. Like function templates, class templates can used for generic class operations and on different data-types.

(b) One difficulty with templates is that they are prone to programming bugs and can be harder to debug than code without templates, producing mystifying compiler errors. On the other hand, templates capturing some generic data structures and tasks can save a huge amount of coding, reducing the scope for errors and improving readability.

(c) Possible measures: First write a specific version with no templates and with typical data types in place of template parameters. Test and debug the code thoroughly before templatising it.

When documenting templates, take care to specify any restrictions on the object types that can be substituted for template parameters.

Use templates for relatively small generic tasks and structures that are likely to be recycled in several different contexts. Larger or more specific tasks might be better off without templates, except perhaps for running time considerations when relevant.

(d)

```
template <typename T>
void Swap(T &n1, T &n2)
{
        T temp;
        temp = n1;
        n1 = n2;
        n2 = temp;
}
```

The output should be:

```
Before calling Swap
i1 = 1, i2 = -1
d1 = -1.2, d2 = -2.5
After calling Swap
```

```
i1 = −1, i2 = 1
d1 = −2.5, d2 = −1.2
```

Answer 3 (30%, each of the 3 subquestions below is worth 5%,20% and 5%).

(a) (Sketch). Use put/call parity and the fact that as the price approaches 0 a put is worth K, and for very large S a put is worthless.

(b) (Sketch). Let $v_{i,j}$ denote the solution at time i and spatial location j. Then we use

$$\begin{aligned}\frac{\partial v(t_i,x_j)}{\partial t} &\approx \frac{v_{i,j}-v_{i-1,j}}{\Delta t}\\ \frac{\partial v(t_i,x_j)}{\partial x} &\approx \frac{v_{i,j+1}-v_{i,j-1}}{2\Delta x}\\ \frac{\partial^2 v(t_i,x_j)}{\partial x^2} &\approx \frac{v_{i,j+1}-2v_{i,j}+v_{i,j-1}}{\Delta x^2}\end{aligned}$$

Using the above formulas into the BSE we obtain,

$$v_{i-1,j} = A_{i,j}v_{i,j-1} + B_{i,j}v_{i,j} + C_{i,j}v_{i,j+1} + D_{i,j}$$

where,

$$\begin{aligned}A_{i,j} &= \frac{\Delta t}{\Delta x}(b_{i,j/2} - a_{i,j}/\Delta x)\\ B_{i,j} &= 1 - \Delta t c_{i,j} + 2\Delta t a_{i,j}/\Delta x)^2\\ C_{i,j} &= -\frac{\Delta t}{\Delta x}(b_{i,j/2} + a_{i,j}/\Delta x)\\ D_{i,j} &= -\Delta t d_{i,j}\end{aligned}$$

(c) Finite difference do not scale well to more than 3-4 dimensions, use Monte Carlo instead.