Question 1 (40%, each of the 4 sub-questions carry equal marks)**.**

1. Consider the following program that attempts to compute the exchange of two currencies.

```
#include <iostream>
 using namespace std;
 int main()
 {
  double USD=100.0;
  double GBP=0;
  GBP = ( USD)*(4/5);
  cout << USD << " USD is " << GBP << " in GBP" << endl;
  return 0;
 }
```

   It produces the following output:"100 USD is 0 in GBP". Can you explain why this happened? How would you modify the code so that it produces the correct output, i.e "100 USD is 80 in GBP".

2. Explain the role of constructors in C++. What is the default constructor? Provide an example of a class with a default constructor, and a constructor that takes a single input.

3. What is the difference between multiple, private and public inheritance in C++? In each case provide an example of when you would use each type of inheritance.

4. Consider the following program.

```
class Base {
private:
    int data;};
class Der1 : public Base { };
class Der2 : public Base {  };
class Join : public Der1, public Der2 {
public:
    void method()
    {
        data = 1;
    }
};
int main()
{
```

```
    Join* j = new Join();
    Base* b = j;
}
```

Can you explain the inheritance structure of the program above (you may use a diagram). The code above will not compile correctly can you suggest a way to fix it?

Question 2 (30%)**.**
In this problem you will be implementing a class which efficiently stores a symmetric matrix. Recall, a d-dimensional symmetric matrix A has the property that $A[i][j] = A[j][i]$.

1. The symmetric matrix class stores only the necessary information, i.e. the diagonal and the upper diagonal entries. Hence, instead of having to store d^2 entries we only store $d(d+1)/2$ entries.

   Consider the following class and fill in the blank lines to complete the class. Refer to the comments above the methods for further details on how they should behave.

   Hint: Remember that $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$ and that $\sum_{i=k}^{n} i = \sum_{i=0}^{n} i - \sum_{i=0}^{k} i$.

```
class SymmetricMatrix{
    private:
        vector<float> M;
    public:
        int dim;

    SymmetricMatrix(vector<float> M, int dim){
        this->M = M;
        this->dim = dim;
    }

    SymmetricMatrix(int dim){
        vector<float> M(dim, 0);
        this->M = M;
        this->dim = dim;
    }

    private:
        // Used internally to convert the provided row and column of the
        // symmetric matrix to an index that can be used to
        // access the corresponding element in this->M.
        int get_idx(int r, int c){
```

```
            if(c < r){
                int tmp = r;
                r = c;
                c = tmp;
            }
            int diag_idx = ________________________________________;
            int offset = ____________________;
            return diag_idx + offset;
        }

    public:
        // Gets the element corresponding to
        // the user-provided row and column.
        float get(int r, int c){
            ________________________________________;
            ________________________________________;
        }

        // Sets the element corresponding to the
        // user-provided row and column to be val.
        void set(float val, int r, int c){
            ________________________________________;
            ________________________________________;
        }
};
```

Question 3 (30%, each of the 3 sub-questions below are worth 5%,10% and 15% respectively)**.**

1. Explain how the Central Limit Theorem (CLT) and the Law of Large Numbers (LLN) can be used to justify the Monte Carlo method.

2. Describe how the CLT and the LLN can be used to derive an appropriate error estimate for the Monte Carlo method. How would you describe the error estimate to a colleague that does not know about Monte Carlo methods or any advanced statistics?

3. Explain the technique of antithetic variables for variance reduction of the following expectation:

   $$\mathbb{E}[f(X)]$$

   where X is a normal random variable with mean 0 and variance 1. What properties should the function f have for the technique to be effective? Justify your answer.

Answer 1.

1. The problem is the line GBP = (USD)*(4/5); it should be GBP = (USD)*(4.0/5.0). C++ does not do automatic conversion of types so 4/5 is represented as an integer, in this case it is 0, i.e. division with integers in C++ returns an integer.

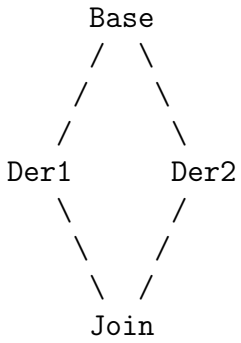
2. The constructor is a member function that is executed automatically whenever an object is created. Constructors have the same name as the class of which they are members, no return type is used for constructors. The default constructor does nothing.

```
class A{
 private:
  int val;
 public:
  A(int x){                //one argument constructor
    val=x;
  }
  A(){                       //default argument constructor
  }
}
int main(){
 A a(3);
 A b();

 return 0;
}
```

3. (sketch) Public inheritance makes public members of the base class public in the derived class. private inheritance makes the public and protected members of the base class private in the derived class. Multiple inheritance is when a class inherits from more than one class. The students are also expected to provide examples.

4. The structure looks like a diamond:



Base is inherited twice, which means any data members declared in Base, such as data above, will appear twice within a Join object. This can create ambiguities: which data did you want to change? For the same reason the conversion from Join* to Base*, or from Join& to Base&, is ambiguous. To resolve the ambiguities instead of saying data = 1 you could say Der2::data = 1, or you could convert from Join* to a Der1* and then to a Base*.

Answer 2.

```
class SymmetricMatrix{
private:
    vector<float> M;
public:
    int dim;

SymmetricMatrix(vector<float> M, int dim){
    this->M = M;
    this->dim = dim;
}

SymmetricMatrix(int dim){
    vector<float> M(dim, 0);
    this->M = M;
    this->dim = dim;
}

private:
    int get_idx(int r, int c){
        if(c < r){
            int tmp = r;
            r = c;
            c = tmp;
        }
        int diag_idx = dim * (dim + 1)/2 - (dim - r) * ((dim - r) + 1)/2;
        int offset = c - r;
        return diag_idx + offset;
    }

public:
    float get(int r, int c){
        int idx = get_idx(r, c);
        return M[idx];
    }

    void set(float val, int r, int c){
        int idx = get_idx(r, c);
```

```
            M[idx] = val;
        }

        void print(){
            for(int r=0; r < dim; r++){
                for(int c=0; c < dim; c++){
                    cout << get(r, c) << " ";
                }
                cout << endl;
            }
        }

};

SymmetricMatrix multiply(int dim, SymmetricMatrix A, SymmetricMatrix B){
    SymmetricMatrix C = SymmetricMatrix(dim);

    for(int r = 0; r < dim; r++){
        for(int c = 0; c < dim; c++){
            float curr_sum = 0;
            for(int k = 0; k < dim; k++){
                curr_sum += A.get(r, k) * B.get(k, c);
            }
            C.set(curr_sum, r, c);
        }
    }
    return C;
}
```

Answer 3 (30%, each of the 3 sub-questions below are worth 5%,10% and 15% respectively)**.**

1. The (CLT) says that for $X_i,\ i = 1, \ldots, N$ i.i.d with mean a and variance b^2

$$\lim_{N\to\infty} \mathbb{P}\left(\alpha \leq \frac{\sum_{i=1}^{N} X_i - Na}{\sqrt{N}b} \leq \beta \right) = \frac{1}{\sqrt{2\pi}} \int_{\alpha}^{\beta} e^{-\frac{1}{2}x^2} dx$$

   So if we take $\alpha = -N^{-1/4}$ and $\beta = N^{1/4}$ we get

$$\mathbb{P}\left(\frac{-b}{N^{1/4}} \leq \frac{1}{N}\sum_{i=1}^{N} X_i - a \leq \frac{b}{N^1/4} \right) = \frac{1}{\sqrt{2\pi}} \int_{-N^{1/4}}^{N^{1/4}} e^{-\frac{1}{2}x^2} dx \to 0 \text{ as } N \to \infty$$

   Thus the CLT tells us that the random variable $\frac{\sum_{i=1}^{N} X_i - Na}{\sqrt{N}b}$ is Gaussian for a large enough N. The LLN then says that,

$$\mathbb{P}\left(\lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} X_i = a \right) = 1$$

   the empirical mean will converge to the real mean. Thus the CLT tells us the distribution of the empirical mean (which enables us to compute error estimates), and the LLN tells us about the fluctuations around the mean (which enables us to conclude that the Monte Carlo method will converge).

2. (sketch) The CLT tells us the distribution of the empirical mean (which enables us to compute error estimates), and the LLN tells us about the fluctuations around the mean (which enables us to conclude that the Monte Carlo method will converge). We know that if $Y \sim N(\mu, \sigma^2)$ then $\mathbb{P}(\left|\frac{Y-\mu}{\sigma}\right| \leq 1.96) \approx 0.95$ i.e. 95% of normal samples lie within two standard deviations (approx). The CLT suggests that for large N the sample mean a_N is approximately $\sim N(a, b^2/N)$ i.e.

$$\mathbb{P}(a_N - 1.96b/\sqrt{N} \leq a \leq a_N + 1.96b/\sqrt{N}) \approx 0.95$$

   We do not know b in practice so we tend to use the sample variance instead and regard the interval:

$$\left[a_N - 1.96b_N/\sqrt{N}, a_N + 1.96b_N/\sqrt{N} \right]$$

as the 95% confidence interval for the true mean. i.e. if we ran MC simulation for a large number of times then we expect 95% of the estimates to lie in this interval

3. The antithetic method in this case is given by,

$$\hat{V}_N = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(f(X_i) + f(-X_i)), \text{ where } X_i \text{ is i.i.d from } N(0,1)$$

We can calculate the

$$\begin{aligned}\text{var}\left(\frac{f(X_i) + f(1-X_i)}{2}\right) &= \frac{1}{4}\text{var}(f(X_i)) + \frac{1}{4}\text{var}(f(1-X_i)) + \frac{1}{2}\text{cov}(f(X_i), f(1-X_i)) \\ &= \frac{1}{2}\text{var}(f(X_i)) + \frac{1}{2}\text{cov}(f(X_i), f(1-X_i))\end{aligned}$$

since X_i and $1 - X_i$ are $N(0,1)$

The success of the method depends on making this covariance term as negative as possible. Thus if f is monotonic then antithetic variables are likely to be useful