سوال 1:

Move semantics :

با استفاده از ریفرینس میتوان به جای کپی کردن متغیر میتوان فقط یک متغیر دیگری ایجادکرد که دقیقا به همان آدرس متغیر اول اشاره میکند و در واقع فقط اسم آن متفاوت است. این کار مزایای زیادی دارد برای مثال میتوان ازکپی کردن کد جلوگیری کرد و یا میتوان متغیر دوم را به صورت const تعریف کرد که در این صورت نمیتوان از طریق دومی اولی را نیز تغییر داد و بسیاری کاربردهای دیگر مثلا در ارسال به تابع یا حتی درکلاس ها برای نمونه اگر در داخل کلاس ما متغیر های داینامیک داشته باشیم باید حتما توابع move constructor و destructor و operator= را تعریف کنیم. چون در غیر این صورت متغیر های داینامیک را چند بار حذف می کند و باعث segmentation error میشود.

Polymorphism :

چند ریختی یعنی این که چند تا نوع یا type متفاوت رابط یکسانی داشته باشند و بشود به شکل یکسانی به آن ها دسترسی داشت.

چندریختی ها به منظور استفاده از توابع کلاسهایی که از هم ارث برده اند استفاده میشود. برای مثال وقتی ما پوینتری ازکلاس والد تعریف میکنیم که به کلاس بچه اشاره میکند در این صورت میتوان از توابع هر دو استفاده کرد که برای کاپپایلر از توابع والد استفاده می کند برای جلوگیری از این موضوع میتوان آن تابعی که مد نظر هست را به صورت virtual تعریف کرد. همچنین در موقع destruct نیز اشتباها destruct میکند و و فقط والد را destruct میکندکه برای همین باید destructorها را نیز به صورت virtual تعریف کرد.

چندریختی جاهایی استفاده میشود که نیاز باشد بصورت generic کد بنویسیم (یعنی این که یک کد خاص با انواع متقاوت کار بکند)

pure abstract:

اگر در داخل یک کلاس فقط یا حداقل یکی از توبع virtual را برابر صفر قرار دهیم مانند: virtual void print() = 0; در این صورت آن کلاس abstract میشود که در این صورت به صورت مستقیم نمیشود از آن کلاس آبجکتی درست کرد البته میتوان به صورت پوینتری یا رفرنسی و استفاده از آبجکتهای کلاس مشتق شده(کلاس بچه) استفاده کرد. برای همین اکثر مواقع کلاس والد را که بقیه کلاس ها از آن مشتق گرفته اند را به این صورت تعریف میکنند مانند سوال 2 این سری تمرینات.

Override :

این کلیدواژه زمانی کاربرد پیدا میکندکه ما کلاس هایی داشته باشیم که از هم ارث ببرند و داخل این کلاس ها توابع virtual باشد در این صورت با گذاشتن کلیدواژه override به کامپایلر یادآوری میکنیم که ما در این توابع از virtual استفاده کرده ایم و در واقع خود کامپایلر این را چک میکند و اگر اشتباهی صورت گفته باشدکامپایلر اررور میدهد. برای مثال اشتباه میتواند به این صورت باشدکه ما درکلاس والد مثلا این تابع را داشته باشیم : virtual void func() و درکلاس مشتق میخوایم تابعی تعریف کنیم که با تابع کلاس والد یکی باشد اما اشتباهی صورت میگیرد مثلا void func(int a) در این صورت کامپایلرکد را درست اجرا میکند ولی این کدی که اجرا میکند منظور ما نیست برای همین به این صورت تعریف میکنیم void func(int a) override تا کامپایلر حواسش باشد.

Inline :

وقتی که تابع inline تعریف شده باشدکامپایلر به صورت مستقیم کد داخل تابع را داخل main برنامه فرار میدهد و اگرتابع inline نباشد از call برای صدا زدن procedure استفاده میکند ، در این صورت تابع سریعتر اجرا میشود.

باید توجه کرد که در تابعی که inline شده علاوه بر این که عملیات push و pop انجام نمیشود به این دلیل که وقتی دستور call داخل اسمبلی استفاده میشود cpu مجبور به صبر میشود تا دستورکاملا اجرا شده و بعد بقیه دستورات را اجراکند حالا اگر مثلا مقدار بازگشتی از تابع داخل یک دستور شرطی استفاده شده بود و ما call نداشته باشیم می تواند با استفاده از branch prediction نتیجه دستورات شرطی تابع را پیش بینی کرده و قسمت های بعدی کد را هم همزمان با تابع اجراکندکه تاثیر خیلی زیادی در سرعت کاپایل خواهد داشت.

البته Inline کردن توابع حجم برنامه را زیاد می‌کند پس فقط توابعی که طول کوتاهی دارند و دفعات زیادی هم استفاده نمیشنوند را inline میکنند.

Explicit :

explicit برای سازنده کلاس تعریف میشه که کامپایلر مقید میشود که فقط و فقط همان نوع داده ای را قبول کندکه در تعریف آمده و در غیر اینصورت در هنگام کامپایل خطا صادر می‌کند. برای مثال:

```
class Complex
{
  Complex(double r = 0.0, double i = 0.0) : real(r), imag(i) {}

  // A method to compare two Complex numbers
  bool operator == (Complex rhs)
};

int main()
{
  // a Complex object
  Complex com1(3.0, 0.0);

  if (com1 == 3.0)
}
```

درست اجرا میشود اما کد زیر با ارور روبرو میشود:

```
explicit Complex(double r = 0.0, double i = 0.0) : real(r), imag(i) {}
```

ارور:

```
no match for 'operator==' in 'com1 == 3.0e+0'
```

برای رفع این مشکل میتوان به صورت زیر عمل کرد:

```
if (com1 == (Complex)3.0)
```