```
// Abstract1.cpp
// Slicing using =
#include <iostream.h>

class B
{ public:
  virtual int get_val() {return 0;};
};

class D1 : public B
{ public:
  D1(int x_val): x(x_val) {};
   virtual int get_val() {return x;};
 private:
  int x;
};

class D2 : public B
{ public:
  D2(int y_val): y(y_val) {};
   virtual int get_val() {return y*y;};
 private:
  int y;
};

main()
{ const int MAX =5;
  D1 Zero(0), One(1), Two(2);
  D2 Three(3), Four(4);

  B B_Array[MAX];
  B_Array[0] = Zero; B_Array[1] = One;
  B_Array[2] = Two;  B_Array[3] = Three;
  B_Array[4] = Four;

  for(int i =0; i < MAX; i++)
    cout << i <<": " << B_Array[i].get_val() << endl;}
```

```
// Abstract2.cpp
// slicing using pointers
#include <iostream.h>

class B
{public:
  int get_val() {return 0;};
};

class D1 : public B
{public:
  D1(int x_val): x(x_val) {};
  int get_val() {return x;};
 private:
  int x;
};

class D2 : public B
{public:
  D2(int y_val): y(y_val) {};
  int get_val() {return y*y;};
 private:
  int y;
};

main()
{ const int MAX =5;
  D1 Zero(0), One(1), Two(2);
  D2 Three(3), Four(4);

  B * B_Array[MAX];
  B_Array[0] = &Zero; B_Array[1] = &One;
  B_Array[2] = &Two;  B_Array[3] = &Three; B_Array[4] = &Four;

  for(int i =0; i < MAX; i++)
    cout << i <<": " << B_Array[i]->get_val() << endl;
}
```

```
// Abstract3.cpp
// Polymorphism and virtual functions
#include <iostream.h>

class B
{ public:
  virtual int get_val() {return 0;};
};

class D1 : public B
{ public:
  D1(int x_val): x(x_val) {};
   virtual int get_val() {return x;};
 private:
  int x;
};

class D2 : public B
{ public:
  D2(int y_val): y(y_val) {};
   virtual int get_val() {return y*y;};
 private:
  int y;
};

main()
{ const int MAX =5;
  D1 Zero(0), One(1), Two(2);
  D2 Three(3), Four(4);

  B * B_Array[MAX];
  B_Array[0] = &Zero; B_Array[1] = &One;
  B_Array[2] = &Two;  B_Array[3] = &Three; B_Array[4] = &Four;

  for(int i =0; i < MAX; i++)
    cout << i <<": " << B_Array[i]->get_val() << endl;
}
```

```
// Abstract4.cpp
// Polymorphism and PURE virtual functions
#include <iostream.h>

class B
{ public:
  virtual int get_val()=0;
};

class D1 : public B
{ public:
  D1(int x_val): x(x_val) {};
   virtual int get_val() {return x;};
 private:
  int x;
};

class D2 : public B
{ public:
  D2(int y_val): y(y_val) {};
   virtual int get_val() {return y*y;};
 private:
  int y;
};

main()
{ const int MAX =5;
  D1 Zero(0), One(1), Two(2);
  D2 Three(3), Four(4);

  B * B_Array[MAX];
  B_Array[0] = &Zero; B_Array[1] = &One;
  B_Array[2] = &Two;  B_Array[3] = &Three; B_Array[4] = &Four;

  for(int i =0; i < MAX; i++)
    cout << i <<": " << B_Array[i]->get_val() << endl;
}
```