Static Scoping vs Dynamic Scoping Example 1

```
1:  int x;
2:  int main() {
3:     x = 14;
4:     f();
5:     g();
6:  }
7:  void f() {
8:     int x = 13;
9:     h();
10: }
11: void g() {
12:    int x = 12;
13:    h();
14: }
15: void h() {
16:    printf("%d\n",x);
17: }
```

Static Scoping vs Dynamic Scoping Example 1

```
1:  int x;
2:  int main() {
3:     x = 14;
4:     f();
5:     g();
6:  }
7:  void f() {
8:     int x = 13;
9:     h();
10: }
11: void g() {
12:    int x = 12;
13:    h();
14: }
15: void h() {
16:    printf("%d\n",x);
17: }
```

Solution:
Static Scoping:
14
14

Static Scoping vs Dynamic Scoping Example 1

```
1:  int x;
2:  int main() {
3:     x = 14;
4:     f();
5:     g();
6:  }
7:  void f() {
8:     int x = 13;
9:     h();
10: }
11: void g() {
12:    int x = 12;
13:    h();
14: }
15: void h() {
16:    printf("%d\n",x);
17: }
```

Solution:
Static Scoping:
14
14

Dynamic Scoping:
13
12

Static Scoping vs Dynamic Scoping Example 2

```
1:  int x;
2:  int main() {
3:     x = 14;
4:     f();
5:     g();
6:  }
7:  void f() {
8:     x = 13;
9:     h();
10: }
11: void g() {
12:    x = 12;
13:    h();
14: }
15: void h() {
16:    printf("%d\n",x);
17: }
```

Static Scoping vs Dynamic Scoping Example 2

```
1:  int x;
2:  int main() {
3:     x = 14;
4:     f();
5:     g();
6:  }
7:  void f() {
8:     x = 13;
9:     h();
10: }
11: void g() {
12:    x = 12;
13:    h();
14: }
15: void h() {
16:    printf("%d\n",x);
17: }
```

Solution:
Static Scoping:
13
12

Static Scoping vs Dynamic Scoping Example 2

```
1:  int x;
2:  int main() {
3:     x = 14;
4:     f();
5:     g();
6:  }
7:  void f() {
8:     x = 13;
9:     h();
10: }
11: void g() {
12:    x = 12;
13:    h();
14: }
15: void h() {
16:    printf("%d\n",x);
17: }
```

Solution:
Static Scoping:
13
12

Dynamic Scoping:
13
12

Static Scoping vs Dynamic Scoping Example 3

```
1:  const int b = 5;
2:  int foo(){
3:      int a = b + 5;
4:      return a;
5:  }
6:  int bar(){
7:      int b = 2;
8:      return foo();
9:  }
10: int main(){
11:     foo();
12:     bar();
13:     return 0;
14: }
```

Static Scoping vs Dynamic Scoping Example 3

```
1:  const int b = 5;
2:  int foo(){
3:     int a = b + 5;
4:     return a;
5:  }
6:  int bar(){
7:     int b = 2;
8:     return foo();
9:  }
10: int main(){
11:    foo();
12:    bar();
13:    return 0;
14: }
```

Solution:
Static Scoping:
foo returns 10
bar returns 10

Static Scoping vs Dynamic Scoping Example 3

```
1:  const int b = 5;
2:  int foo(){
3:     int a = b + 5;
4:     return a;
5:  }
6:  int bar(){
7:     int b = 2;
8:     return foo();
9:  }
10: int main(){
11:    foo();
12:    bar();
13:    return 0;
14: }
```

Solution:
Static Scoping:
foo returns 10
bar returns 10

Dynamic Scoping:
foo returns 10
bar returns 7

Static Scoping vs Dynamic Scoping Example 4

```
1:  x=1;
2:  function g () {
3:    echo $x ;
4:    x=2 ;
5:  }
6:  function f () {
7:    local x=3;
8:    g;
9:  }
10: f;
11: echo $x;
```

Static Scoping vs Dynamic Scoping Example 4

```
1:  x=1;
2:  function g () {
3:    echo $x ;
4:    x=2 ;
5:  }
6:  function f () {
7:    local x=3;
8:    g;
9:  }
10: f;
11: echo $x;
```

Solution:
Static Scoping:
1
2

Static Scoping vs Dynamic Scoping Example 4

```
1:  x=1;
2:  function g () {
3:    echo $x ;
4:    x=2 ;
5:  }
6:  function f () {
7:    local x=3;
8:    g;
9:  }
10: f;
11: echo $x;
```

Solution:
Static Scoping:
1
2
Dynamic Scoping:
3
1

Static Scoping vs Dynamic Scoping Example 5

```
1:  n:integer
2:  procedure first
3:    n:=1
4:  procedure second
5:    n:integer
6:    first()
7:  n:=2
8:  if read_integer() > 0
9:    second();
10: else
11:   first();
12: write_integer(n)
```

Static Scoping vs Dynamic Scoping Example 5

```
1:  n:integer
2:  procedure first
3:    n:=1
4:  procedure second
5:    n:integer
6:    first()
7:  n:=2
8:  if read_integer() > 0
9:    second();
10: else
11:   first();
12: write_integer(n)
```

Solution:
Static Scoping:
1

Static Scoping vs Dynamic Scoping Example 5

```
1:  n:integer
2:  procedure first
3:    n:=1
4:  procedure second
5:    n:integer
6:    first()
7:  n:=2
8:  if read_integer() > 0
9:    second();
10: else
11:   first();
12: write_integer(n)
```

Solution:
Static Scoping:
1
Dynamic Scoping:
It depends on the value given for read_integer(). If the input is positive then it is 2 otherwise 1