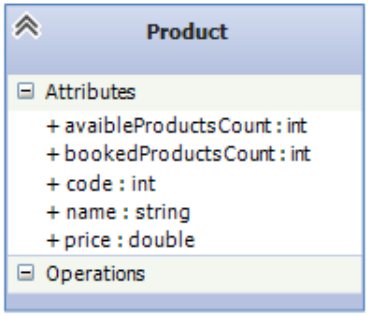
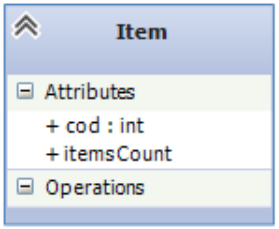
WHITE BOX EXERCISES

Basis Path Technique 2 session

Exercise 3

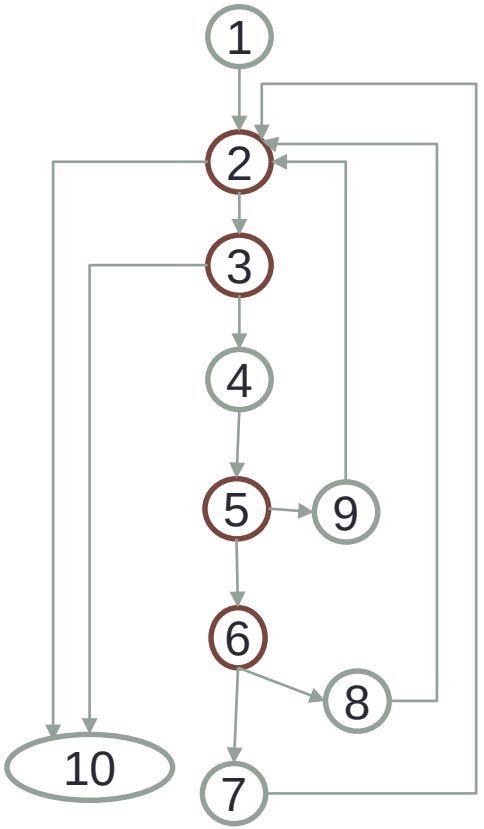
```
static public int bookItems(ArrayList products, Item item, out double cost, out string message)
{
    int j;
    Product product;
    j = 0;
    message = "Product not found";
    cost = 0;

    while ((j < products.Count ) && (message.Equals("Product not found")))
    { product = products[j] as Product;
        if (item.code == product.code)
        { if (item.itemsCount <= product.availbleProductsCount)
            {
                cost = cost + item.itemsCount * product.price;
                product.availbleProductsCount -= item.itemsCount;
                product.bookedProductsCount += item.itemsCount;
                message = "Product booked";
            }
            else
            {
                message = "Not enough products";
            }
        }
        else
        {
            j++;
        }
    }
    return j;
}
```





Exercise 3

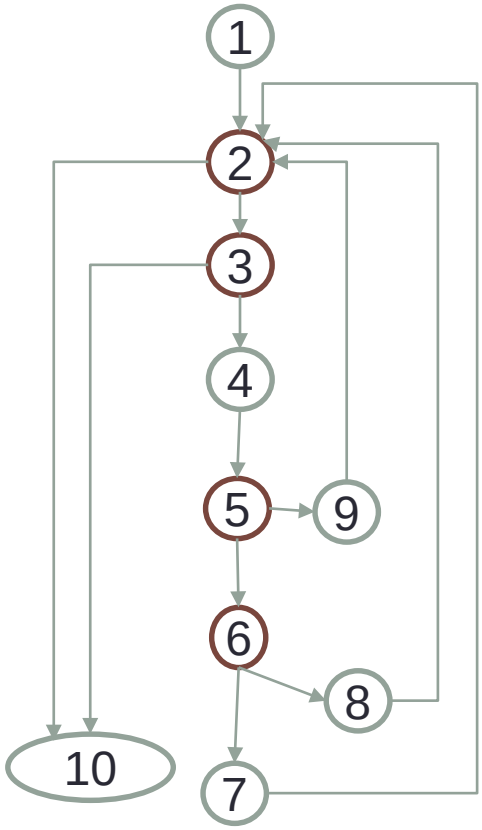
```
static public int bookItems(ArrayList products, Item item, out double cost, out string message)
{
    int j;
    Product product;
    j = 0;
    message = "Product not found";
    cost = 0;

    while ((j < products.Count ) && (message.Equals("Product not found")))
    { product = products[j] as Product;
        if (item.code == product.code)
        { if (item.itemsCount <= product.availableProductsCount)
            {
                cost = cost + item.itemsCount * product.price;
                product.availableProductsCount -= item.itemsCount;
                product.bookedProductsCount += item.itemsCount;
                message = "Product booked";
            }
            else
            {
                message = "Not enough products";
            }
        }
        else
        {
            j++;
        }
    }
    return j;
}
```



V(G)= 5
Areas = 5
Predicate Nodes = 4 → 4+1=5
Nodes = 10 → 13-10+2 =5
Edges = 13

Exercise 3

Path
{1,2,10} No products
{1,**2,3**,10}
{1,2,**3,4**,5,9,2,10} Only one product, not desired
{1,2,3,4**,5,6**,8,2,10} Only one product, desired but with not enough stock
{1,2,3,4,5,**6,7**,2,310} Only one product. Desired product with enough stock



Exercise 3

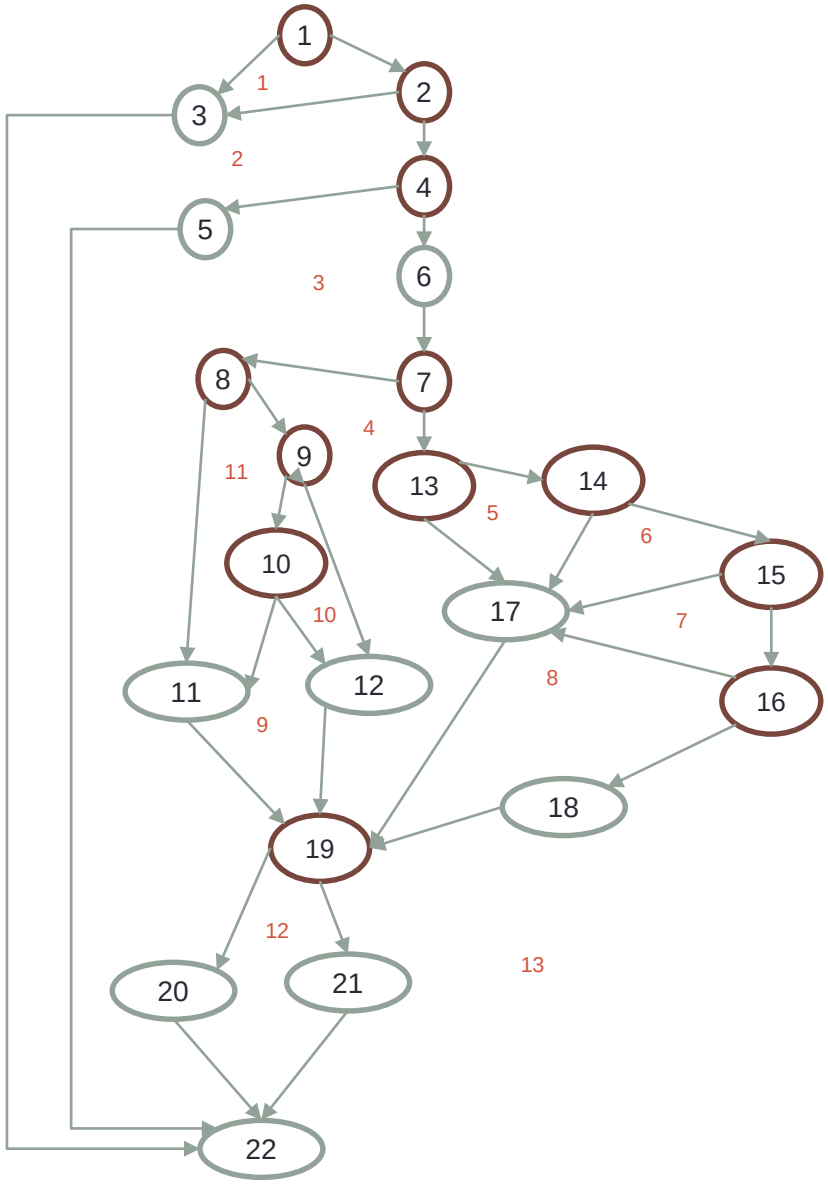
Path	Input		Output			
,	Products	Items	Return	Cost	Message	Products.out
{1,2,10} No products	[ ]	{code=2; itemsCount=5}	0	0,0	Product not found	same
{1,2,3,10}	No possible		No possible			
{1,2,3,4,5,9,2,10} Only one product, not desired	[{code = 5; availableProductsCount =5; bookedProductsCount = 5; price = 10}]	{code=2; itemsCount=10 }	1	0,0	Product not found	same
{1,2,3,4,5,6,8,2,10} Only one product, desired but with not enough stock	[{code = 2; availableProductsCount = 5; bookedProductsCount = 5; price = 10}]	{code=2; itemsCount=10 }	0	0,0	Not enough products	same
{1,2,3,4,5,6,7,2,3,10} Only one product. Desired product with enough stock	[{code = 2; availableProductsCount =15; bookedProductsCount = 5; price = 10}]	{code=2; itemsCount=10 }	0	100	Product booked	[{code = 2; availableProductsCount =5; bookedProductsCount = 15; price = 10}]

Exercise 4

```
static public int valid_date(int dd, int mm, int yy)
{
    if (mm < 1 || mm > 12)
    {
        return 0;
    }
    if (dd < 1)
    {
        return 0;
    }

    int days;
    if (mm == 2)
    {
        // leap year
        if (yy % 400 == 0 || (yy % 4 == 0 && yy % 100 != 0))
        {
            days = 29;
        }
        else days = 28;
    }
    else if (mm == 4 || mm == 6 || mm == 9 || mm == 11)
    {
        days = 30;
    }
    else days = 31;

    if (dd > days)
    {
        return 0;
    }
    return 1;
}
```

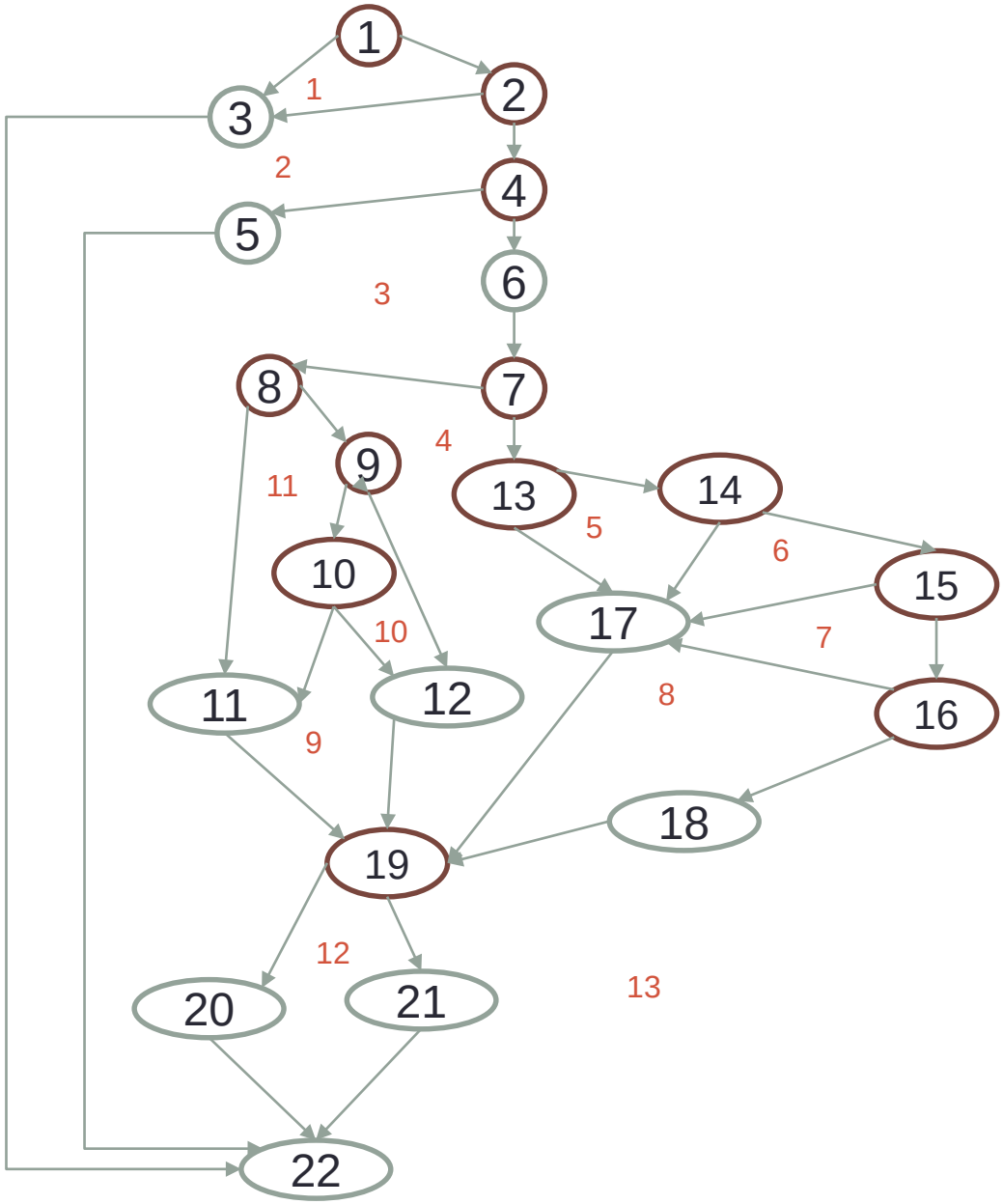
Exercise 4

```
static public int valid_date(int dd, int mm, int yy)
{
    if (mm < 1 || mm > 12)
    {
        return 0;
    }
    if (dd < 1)
    {
        return 0;
    }

    int days;
    if (mm == 2)
    {
        // leap year
        if (yy % 400 == 0 || (yy % 4 == 0 && yy % 100 != 0))
        {
            days = 29;
        }
        else days = 28;
    }
    else if (mm == 4 || mm == 6 || mm == 9 || mm == 11)
    {
        days = 30;
    }
    else days = 31;

    if (dd > days)
    {
        return 0;
    }
    return 1;
}
```



Exercise 4

V(G)= 13
Areas = 13
Predicate Nodes = 12 → 12+1=13
Nodes = 22 → 33-22+2 =13
Edges = 33



Exercise 4

	Path	Input	Output
1	{1,3,22} Month <1	mm=-1; dd=any;yy=any	0
2	{1,**2,3**,22} Month>12	mm=-13; dd=any; yy=any	0
3	{1,**2,4**,5,22} Month valid. Days <1	mm=1; dd=-1; yy=any	0
4	{1,2,**4,6**,7,8,11,19,20,22} February, year divisible by 400 (leap). Days>29	mm=2; dd=30; yy=2000;	0
5	{1,2,4,6,**7,13**,17,19,20,22} April, Days>30	mm=4; dd=31; yy=any	0
6	{1,2,4,6,7,**13,14**,17,19,20,22} June, Days>30	mm=6; dd=31; yy=any	0
7	{1,2,4,6,7,13,**14,15**,17,19,20,22} Sept, Days>30	mm=9; dd=31; yy=any	0
8	{1,2,4,6,7,13,14,**15,16**,17,19,20,22} Nov, Days>30	mm=11; dd=31; yy=any	0
9	{1,2,4,6,7,13,14,15,**16,18**,19,20,22} Dic, Days>30	mm=12; dd=32; yy=any	0
10	{1,2,4,6,7,13,14,15,**16,17**,19,21,22} Dic, Days valid	mm=12; dd=31; yy=any	1
11	{1,2,4,6,7**,8,9**,12,19,20,22} February, year not divisible by 400, no divisible by 4. Days>28	mm=2; dd=31; yy=2005	0
12	{1,2,4,6,7,8,**9,10**,11,19,20,22} February. Divisible by 4 and not by 100 (leap). Days>29	mm=2; dd=30; yy=2004	0
13	{1,2,4,6,7,8,**9,10**,11,19,20,22} February. No divisible by 400.Divisible by 4 and by 100 (no leap). Days>28	mm=2; dd=30; yy=2100	0