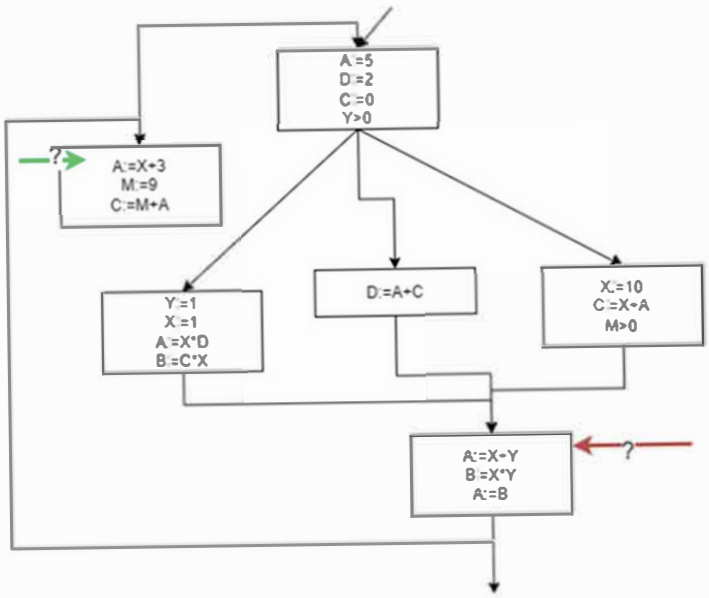
مسئله‌ی ۱.

برنامه زیر را در نظر بگیرید و سپس با توجه به آن به بخش های الف و ب پاسخ دهید:



الف) فرض کنید همه متغیرها در هنگام خروج مرده باشند (فلش سبز). با جرای الگوریتم liveness analysis کدام یکی از متغیرها در نقطه مشخص شده زنده هستند؟

با فرض اینکه فلش از بلاک اول به بلاک سبز است، داریم: (فرض دیگر در صفحه بعد)

با توجه به این که liveness analysis، پایین به بالا انجام می‌شود و بلاک حاوی فلش سبز، هیچ یال خارجی ای ندارد، می‌توان آن را به عنوان پایین‌ترین بلاک ستقیما آنالیز کرد، داریم:

X

true
A := X + 3
false
M := 9
false
C := M + A
false

M

false
A := X + 3
false
M := 9
true
C := M + A
false

A

false
A := X + 3
true
M := 9
true
C := M + A
false

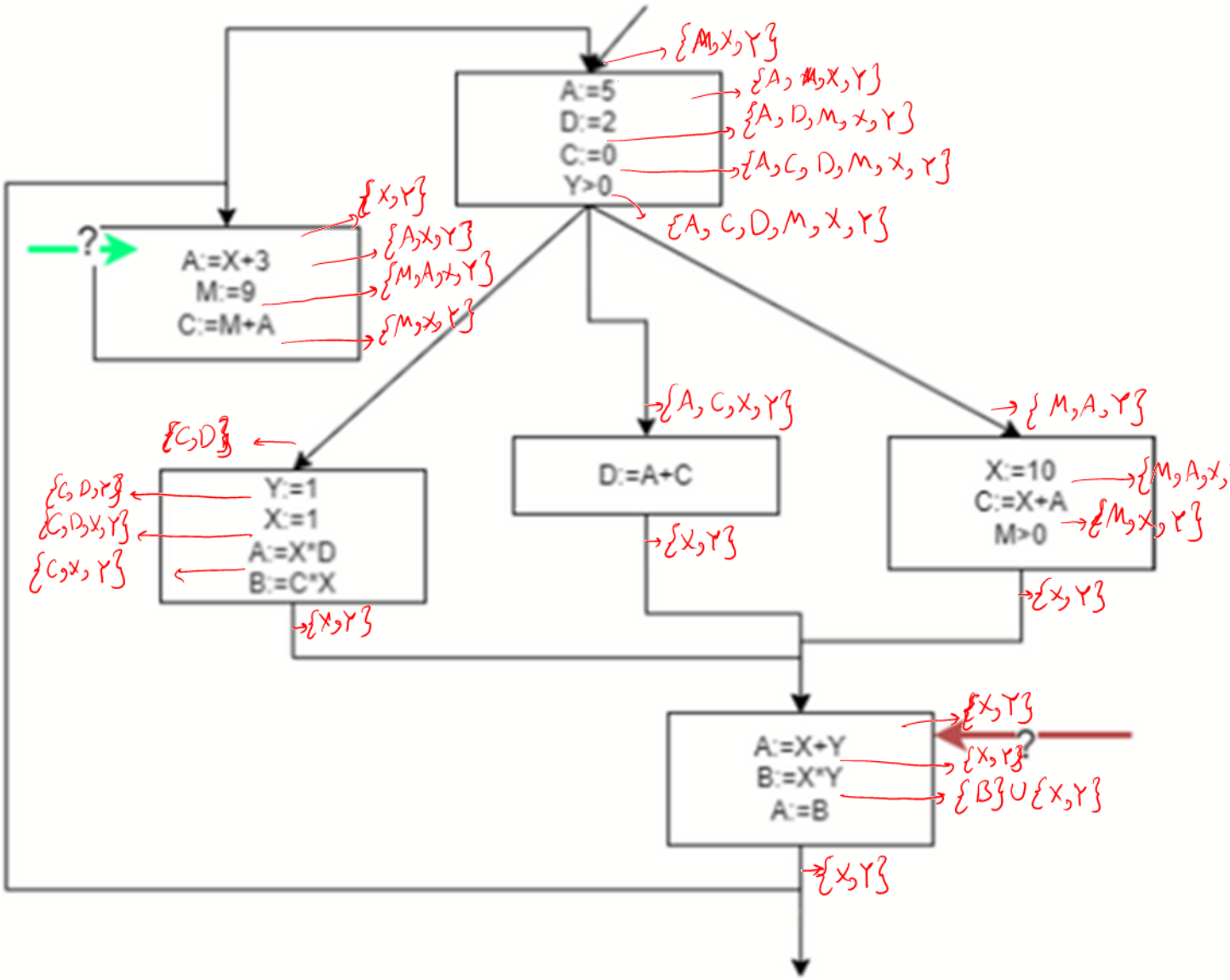
باقی متغیرها

false
A := X + 3
false
M := 9
false
C := M + A
false

بنابراین تنها متغیر زنده در محل فلش سبز، متغیر X است.

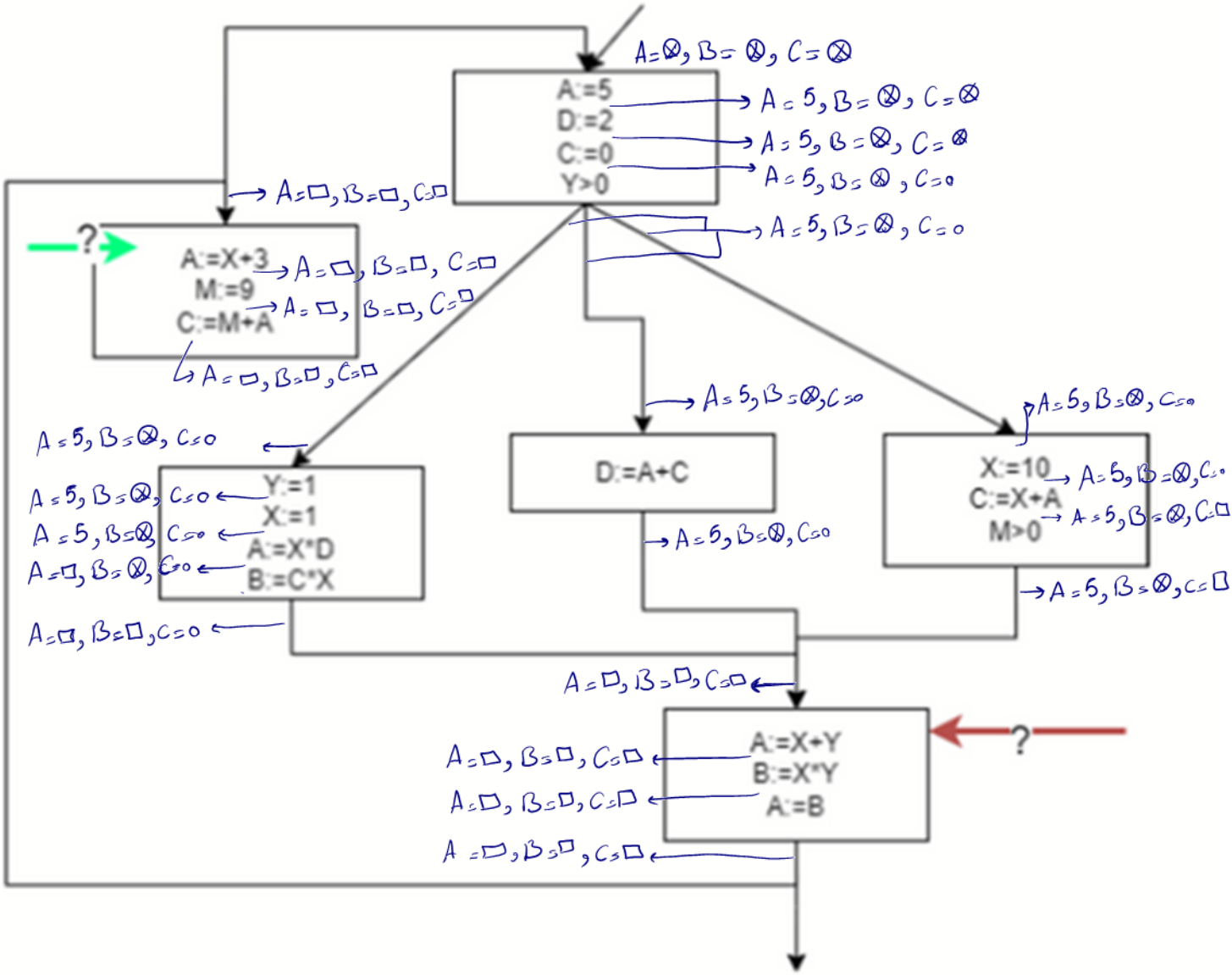
با توجه اینکه فلش از بلاک سبز به بلاک اول است، داریم:

متغیرهای زنده را در هر خط می نویسیم



در نهایت متغیرهای X و Y در هنگام فلش سبز زنده هستند.

ب) با اجرای الگوریتم constant propagation در نقطه ای که با فلش قرمز مشخص شده است information dataflow را برای A و B و C بدست بیاورید.



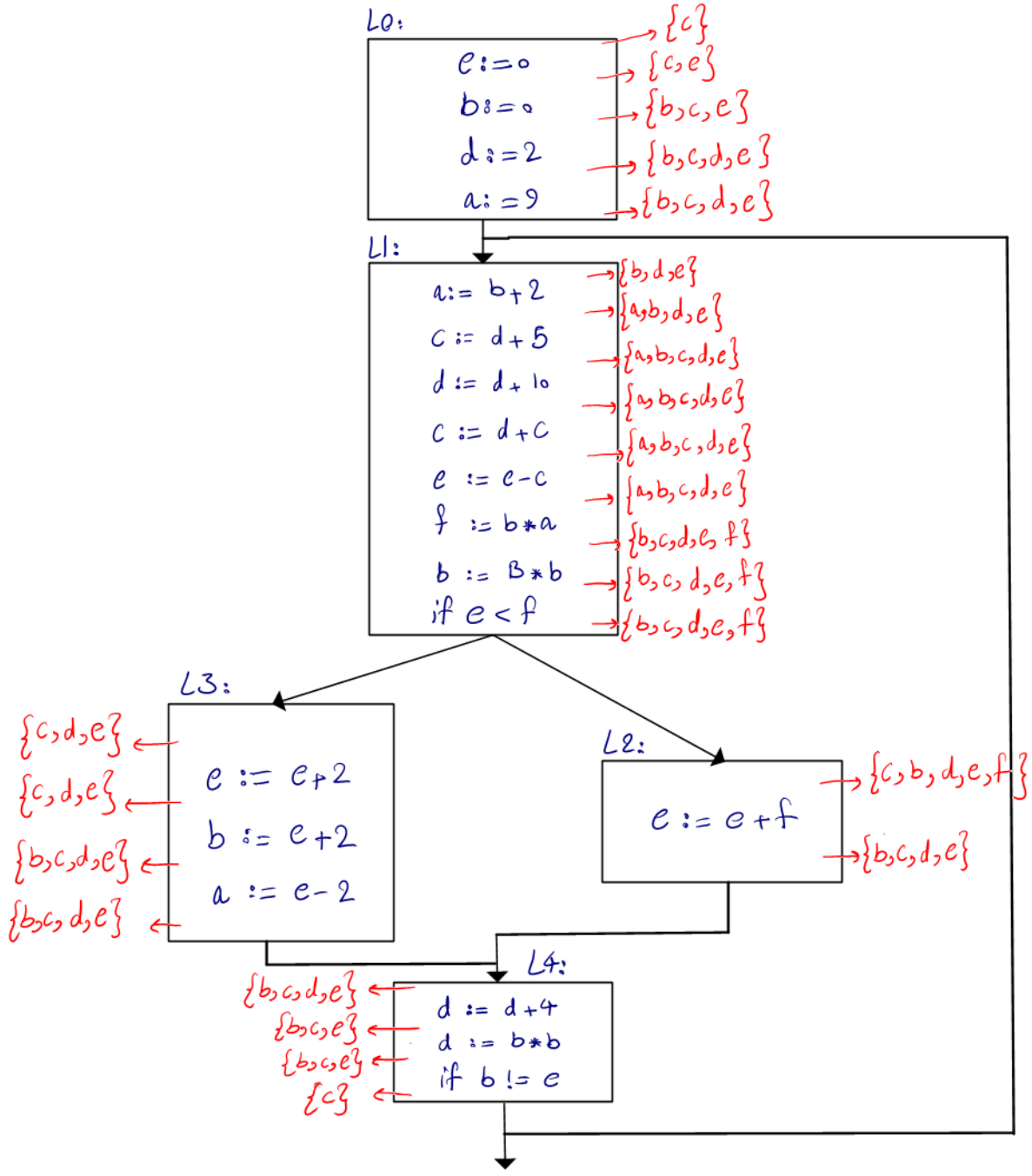
بنابراین در نقطه‌ی مشخص شده با فلش قرمز، وضعیت هر 3 متغیر A و B و C نامشخص (□) است.

مسئله‌ی ۲.

فرض کنید در انتهای قطعه کد زیر فقط متغیر c زنده می‌باشد، ابتدا data flow را رسم کنید و سپس عملیات liveness analysis را با نوشتن مراحل بر روی کد زیر اجرا کنید.

```
L0: e := 0
b := 0
d := 2
a := 9
L1: a := b + 2
c := d + 5
d := d + 10
c := d + c
e := e - c
f := b * a
b := B * b
if e < f goto L3
L2: e := e + f
goto L4
L3: e := e + 2
b := e + 2
a := e - 2
L4: d := d + 4
d:= b * b
if b != e goto L1
```

در هر مرحله متغیرهای زنده را طبق الگوریتم مشخص می‌کنیم



مسئله‌ی ۳.

تکه کد زیر را در نظر بگیرید:

```
int x = 10;
int y = 20;
if (x > y) {
    int z = x + y;
} else {
    int w = x - y;
}
int result = y - x;
```

با استفاده از روش شناسایی و حذف تکه کد مرده بهینه سازی را انجام داده و کد خروجی را بنویسید. برای بهینه سازی خود دلیل نیز بیاورید.

باتوجه به single assignment form بودن کد ابتدا constant propagation انجام می‌دهیم:

```
int x = 10;
int y = 20;
if(10 > 20){
    int z=10 + 20;
}else{
    int w = 10 - 20;
}
int result = 20-10;
```

Constant Folding

```
int x = 10;
int y = 20;
if(10 > 20){
    int z=30;
}else{
    int w=-10;
}
int result =10;
```

removing unreachable blocks

```
int x = 10;
int y = 20;
{int w=-10;}
int result =10;
```

dead code elimination

```
int x = 10;
int y = 20;
int result=10;
```

برای حذف خطوط باقی‌مانده باید liveness analysis انجام شود.

مسئله‌ی ۴.

تکه کد زیر را در نظر بگیرید:

```
int x = 2;
int y = 3;
int z = x + y;
x = 4;
int w = z * 2;
```

با استفاده از روش های بهینه سازی، کد بالا را بهینه کرده و با توضیحات بنویسید. میتوانید از روش Advanced Propagation Constant Global استفاده کنید.

```
int x = 2;
int y = 3;
int z = 2+ 3;
x = 4;
int w = z x 2;
```

constant propagation →

```
int x = 2;
int y = 3;
int z = 5;
x = 4;
int w = z x 2;
```

constant folding →

```
int x = 2;
int y = 3;
int z = 5;
x = 4;
int w = 5x2;
```

constant propagation →

```
int x = 2;
int y = 3;
int z = 5;
x = 4;
int w = 10;
```

constant folding →

```
int y = 3;
int z = 5;
int x = 4;
int w = 10;
```

dead code elimination →

پاسخ: با توجه به استفاده از AGCP، امکان چندبار عمل CP و CF وجود دارد.

مسئله‌ی ۵.

قطعه کد شبه زیر شبیه به برنامه‌های پاسکال است که در آن می‌توان تعاریف تودرتو از رویه‌ها را داشت. حالت symbol table و scope stack را بعد از کامپایل خطوط ۱۰ و ۱۴ به ترتیب نشان دهید. همچنین، آدرس هر متغیر در symbol table (یعنی ویژگی شناسه توکن) را برای این دو عبارت نشان دهید.

```
1  program main();
2     var i, j, s: integer;
3
4     procedure p();
5         var a[1..10] real;
6         function f1(n: integer): integer;
7             var b[1..10] real;
8             procedure p1(s: real);
9                 var a real;
10                 a := s + b[i % 10 + 1];
11            end p1;
12            procedure p2();
13                var arr2[1..3] real;
14                arr2[1] = j + a[n % 5 + 1];
15            end p2;
16        end f1;
17    end p;
18
19 end main;
```

بعد از کامپایل خط ۱۴:

symbol table

	lexeme	proc/func/var	No. args/cell	type	scope
0	main	proc	0	—	1
1	i	var	0	integer	1
2	j	var	0	integer	1
3	s	var	0	integer	1
4	p	proc	0	—	1
5	a	array	10	real	2
6	f1	proc	1	integer	2
7	n	param	0	integer	3
8	b	array	10	real	3
9	p1	proc	1	—	3
10	p2	proc	0	—	3
11	arr2	array	3	real	4

Scope Stack
11
7
5
0

آدرس هر متغیر در symbol table:

arr2: 11 j: 2 a: 5 n: 7

symbol table

	lexeme	proc/func/var	No. args/cell	type	scope
0	main	proc	0	—	1
1	i	var	0	integer	1
2	j	var	0	integer	1
3	s	var	0	integer	1
4	p	proc	0	—	1
5	a	array	10	real	2
6	f1	proc	1	integer	2
7	n	param	0	integer	3
8	b	array	10	real	3
9	p1	proc	1	—	3
10	s	param	0	integer	4
11	a	var	0	real	4

بعد از کامپایل خط 10:

Scope Stack
10
7
5
0

آدرس هر متغیر در symbol table:

a: 11 s: 10 b: 8 i: 1