پاسخ تمرین سری پنجم — محیط‌های زمان اجرا و بهینه‌سازی — موعد تحویل:

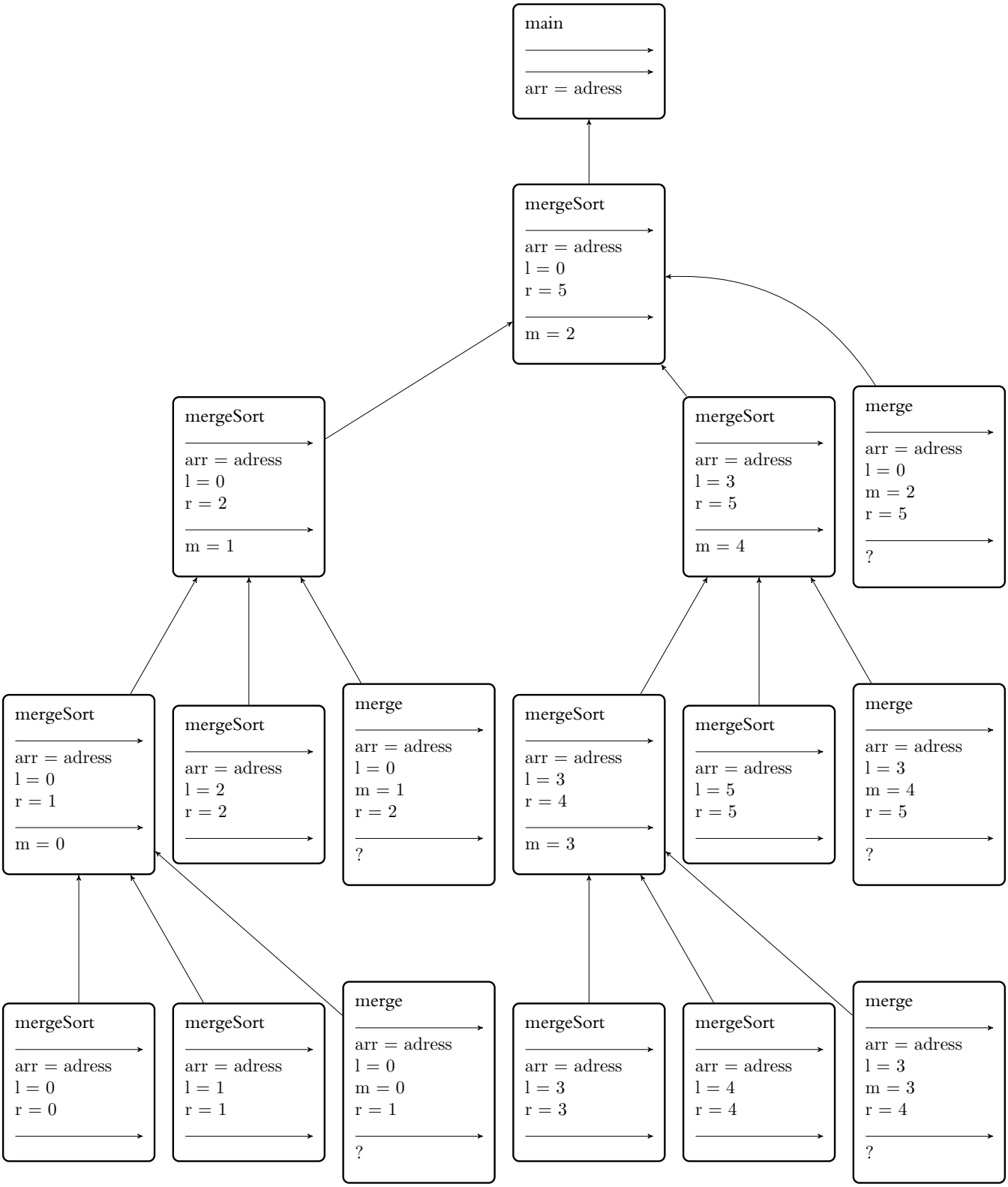
۱. برای برنامه زیر، activation tree را رسم کنید.

ـ فرض کنید record activation برای هر تابع شامل موارد زیر می شود:

- ورودی های تابع
- متغیر های محلی

```
void main(){
        int arr[] = {3, 34, 1, 5, 6, 8};
        mergeSort(arr, 0, arr.length - 1);
}
void mergeSort(int arr[], int l, int r) {
        if (l < r) {
                int m = (l + r) / 2;
                mergeSort(arr, l, m);
                mergeSort(arr, m + 1, r);
                merge(arr, l, m, r);
        }
}
void merge(int arr[], int l, int m, int r)
{
        // Merges arr[l, m] and arr[m + 1, r]
}
```

پاسخ:


main
arr = adress
mergeSort
arr = adress
l = 0
r = 5
m = 2
mergeSort
arr = adress
l = 0
r = 2
m = 1
mergeSort
arr = adress
l = 3
r = 5
m = 4
merge
arr = adress
l = 0
m = 2
r = 5
?
mergeSort
arr = adress
l = 0
r = 1
m = 0
mergeSort
arr = adress
l = 2
r = 2
merge
arr = adress
l = 0
m = 1
r = 2
?
mergeSort
arr = adress
l = 3
r = 4
m = 3
mergeSort
arr = adress
l = 5
r = 5
merge
arr = adress
l = 3
m = 4
r = 5
?
mergeSort
arr = adress
l = 0
r = 0
mergeSort
arr = adress
l = 1
r = 1
merge
arr = adress
l = 0
m = 0
r = 1
?
mergeSort
arr = adress
l = 3
r = 3
mergeSort
arr = adress
l = 4
r = 4
merge
arr = adress
l = 3
m = 3
r = 4
?

۲. برای قطعه کد زیر ساختار Stack Frame را برای هر یک از موارد زیر نشان دهید. (در پشته نشان داده شده تعیین کنید هر مقدار را کدام تابع در پشته push می کند.)

الف) زمانی که اولین بار خط ۱۱ اجرا می شود. (بعد از اجرا شدن)

ب) زمانی که خط ۸ اجرا می شود. (بعد از اجرا شدن)

(مقدار fp را نسبت به fp اولیه تعیین کنید و مقدار ra را با استفاده از شماره خطوط زیر تعیین کنید.)

```
1. void main() {
2.         int a = 24;
3.         int b = 36;
4.         int c = gcd(a, b);
5. }
6. int gcd(x, y) {
7.         if (x == 0){
8.                 return y;
9.         }
10.         int r = b % a;
11.         return gcd(r, a);
12. }
```

پاسخ:

پس از اجرای خط ۱۱ برای اولین بار، StackFrame به صوت زیر در می آید که به ترتیب از خانه ۲ تا ۸ توسط main و از خانه ۹ تا ۱۳ توسط gcd(24,36) در پشته push شده است.

0	fp of caller
1	ra of caller
2	a = 24
3	b = 36
4	c = gcd(24, 36)
5	x = 24
6	y = 36
7	fp of caller = 0 on stack
8	ra of caller = 4
9	r = 12
10	x = 12
11	y = 24
12	fp of caller = 7 on stack
13	ra of caller = 11

خانه ۰ و ۱ مربوط به caller تابع main هستند.

پس از اجرای خط ۸ پشته به شکل زیر:
به ترتیب از خانه ۲ تا ۸ توسط main
از خانه ۹ تا ۱۳ توسط gcd(24, 36)
از خانه ۱۴ تا ۱۸ توسط gcd(12, 24)
و خانه ۱۹ پشته توسط gcd(0, 12) push شده اند.

0	fp of caller
1	ra of caller
2	a = 24
3	b = 36
4	c = gcd(24, 36)
5	x = 24
6	y = 36
7	fp of caller = 0 on stack
8	ra of caller = 4
9	r = 12
10	x = 12
11	y = 24
12	fp of caller = 7 on stack
13	ra of caller = 11
14	r = 0
15	x = 0
16	y = 12
17	fp of caller = 12 on stack
18	ra of caller = 11
19	return value = 12 (c = 12)

۳. خروجی برنامه زیر را در هر یک از شرایط زیر با ذکر دلیل تعیین کنید.

- Call by value
- Call by reference
- Call by name
- Call by value-result

```
int n = 1;
int array[] = {0, 1, 2, 3, 4};

void printAll(int a, int b){
        int t;
        int i = 0;
        t = a;
        a = b;
        b = t;
        print(a, b, i, array);
}

Void main() {
        printAll(i, array[i]);
        print(i, array);
}
```

پاسخ:

Call by value:

```
1 1 0 {0 1 2 3 4}
1 {0 1 2 3 4}
```

Call by reference:

```
1 1 0 {0 1 2 3 4}
1 {0 1 2 3 4}
```

Call by name:

```
0 0 0 {0 1 2 3 4}
1 {0 1 2 3 4}
```

Call by value–result:

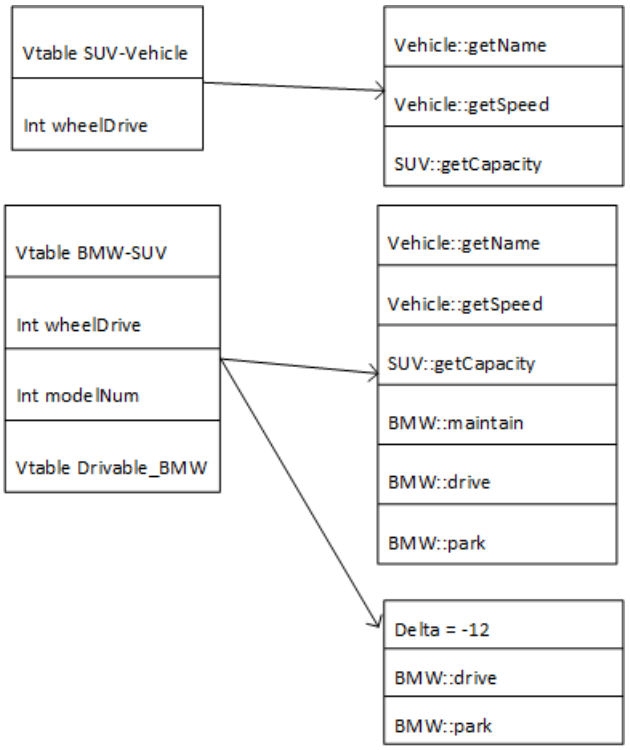
```
1 1 0 {0 1 2 3 4}
1 {0 1 2 3 4}
```

۴. با توجه به کد زیر vtable را برای اشیا زیر رسم کنید.

- SUV s = SUV()
- Drivable b = BMW()

```
Class Vehicle {
        Public void getName();
        Public int getSpeed();
}
Class SUV extends Vehicle {
        Int wheelDrive;
        Public void getCapacity();
}
Interface Drivable {
        Public void drive();
        Public void park();
}
Class BMW extends SUV implements Drivable {
        Int modelNum;
        Public void maintain();
}
```

پاسخ:



۵. برای کد زیر، کد سه آدرس تولید کنید. (پارامتر ها به شکل pass by value به توابع داده می شوند)

```
class A {
        int fieldA;
        int methodA(int a){
                return (a + this.fieldA) * 10;
}
}
class B extends A {
        int fieldB;
        int methodB(int param){
                return this.methodA(6) * (param + this.fieldB);
        }
}
Void main(){
        A a = B();
        b.fieldA = 5;
        b.fieldB = 10;
        int x;
        x = b.methodB(2);
}
```

پاسخ:

```
_A.methodA:
    BeginFunc 8;
    _t0 = a + *(this + 4);
    _t1 = _t0 * 10
    Return _t1;
    EndFunc;
Vtable A = _A.methodA,
;

_B.methodB:
    BeginFunc 24;
    _t0 = 6;
    PushParam _t0;
    PushParam this;
    _t1 = *this;
    _t2 = *_t1;
    _t3 = ACall _t2;
    PopParams 8;
    _t4 = param + *(this + 8);
    _t5 = _t3 * _t4;
```

```
    Return _t5;
    EndFunc;
Vtable B = _A.methodA, _B.methodB
;

main:
    BeginFunc 20;
    _t0 = 12;
    PushParam _t0;
    b = LCall _Alloc;
    PopParams 4;
    _t1 = B;
    *b = _t1;
    _t2 = 5;
    *(b + 4) = _t2;
    _t3 = 10;
    *(b + 8) = _t3;
    _t4 = 2;
    PushParam _t4;
    PushParam b;
    _t5 = *b;
    _t6 = *(_t5 + 4);
    _t7 = ACall _t6;
    PopParams 8;
    x = _t7;
    EndFunc
;
```

در تابع main با توجه به اینکه به نظر یک خطای نگارشی در صورت سوال رخ داده، فرض شده که به جای جمله A a=B(); جمله A b=B(); قرار دارد که به این ترتیب خطای موجود در صورت سوال با کمترین تغییرات رفع میشود.

۶. کد زیر را در نظر بگیرید.

```
a = 1 + 2;
b = a + b;
z = a * 2;
c = b + e;
d = c + b;
x = b + 3;
z = a * 8;
t = c - 2;
f = x + f;
y = x - 2;
d = d - y;
```

فرض کنید بعد از اجرای این کد متغیر های d و z و x زنده اند.

الف) برای هر خط متغیرهای زنده را بنویسید.

ب) با کمک بهینه سازی های محلی که در درس یادگرفته اید سعی کنید این کد را به بهینه ترین حالت ممکن تبدیل کنید.

پاسخ:

الف)

```
    { f, b, e }
a = 1 + 2;
    { f, a, b, e }
b = a + b;
    { f, a, b, e }
z = a * 2;
    { f, a, b, e }
c = b + e;
    { f, c, a, b }
d = c + b;
    { f, c, a, b, d }
x = b + 3;
    { x, f, c, a, d }
z = a * 8;
    { x, z, f, c, d }
t = c - 2;
    { x, z, f, d }
f = x + f;
    { x, z, d }
y = x - 2;
    { x, z, y, d }
d = d - y;
    { x, z, d }
```

ب)

```
a = 1 + 2;
b = a + b;
c = b + e;
d = c + b;
x = b + 3;
z = a * 8;
y = x - 2;
d = d - y;
```

```
b = 3 + b;
c = b + e;
d = c + b;
x = b + 3;
z = 24;
y = x - 2;
d = d - y;
```

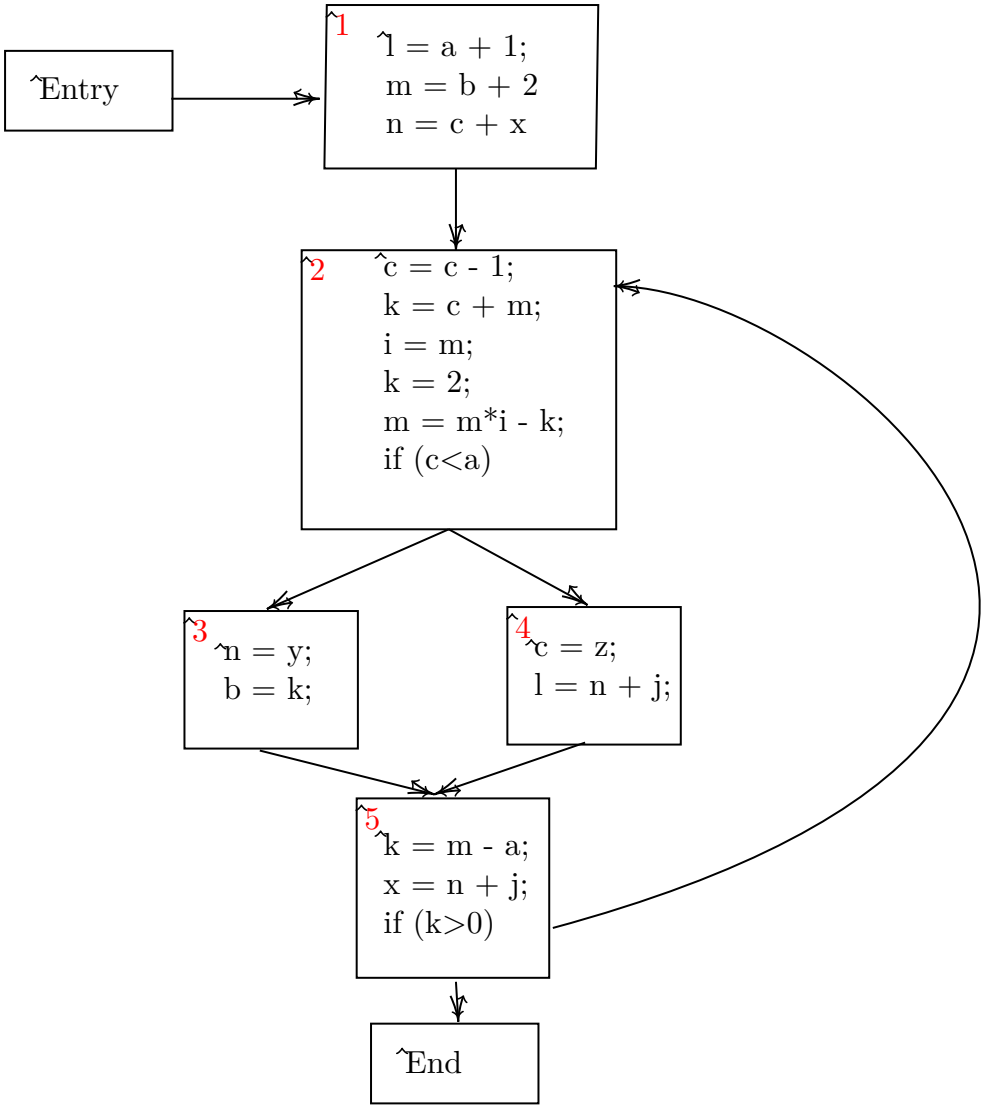
۷. کد زیر را در نظر بگیرید.

```
l = a + 1;
m = b + 2;
n = c + x;
do {
        c = c - 1;
        k = c + m;
        i = m;
        k = 2;
        m = m * i - k;
        if( c < a )
                n = y;
                b = k;
        else
                c = z;
                l = n + j;
        k = m - a;
        x = n + j;
}
while (k > 0);
```

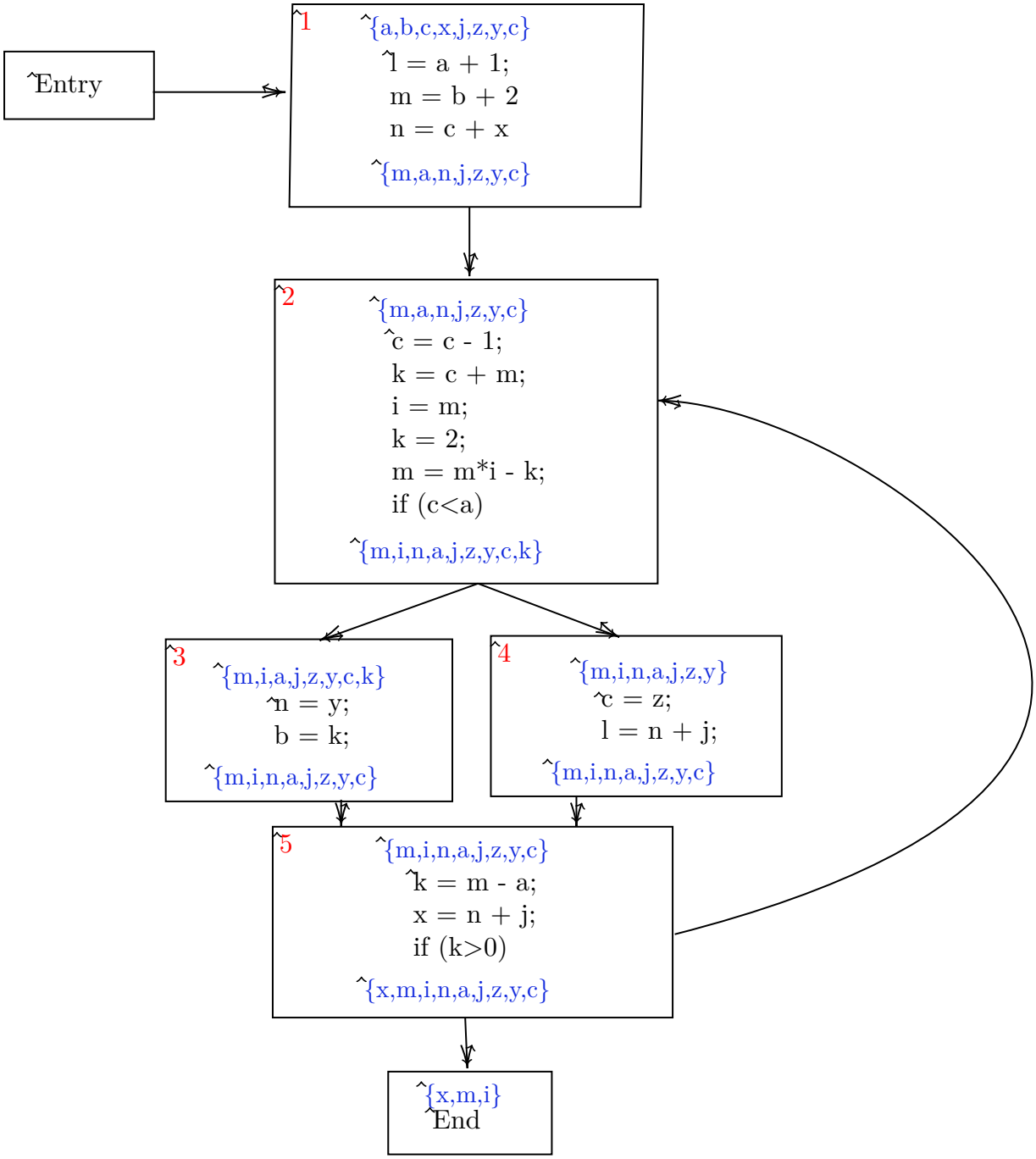
الف) نمودار جریان داده ای را برای آن رسم کنید.
ب) تحلیل متغیر های زنده را برای این کد انجام دهید.
ج) انتشار سراسری ثابت انجام دهید.
د) افزونگی های جزئی را حذف کنید.

پاسخ:

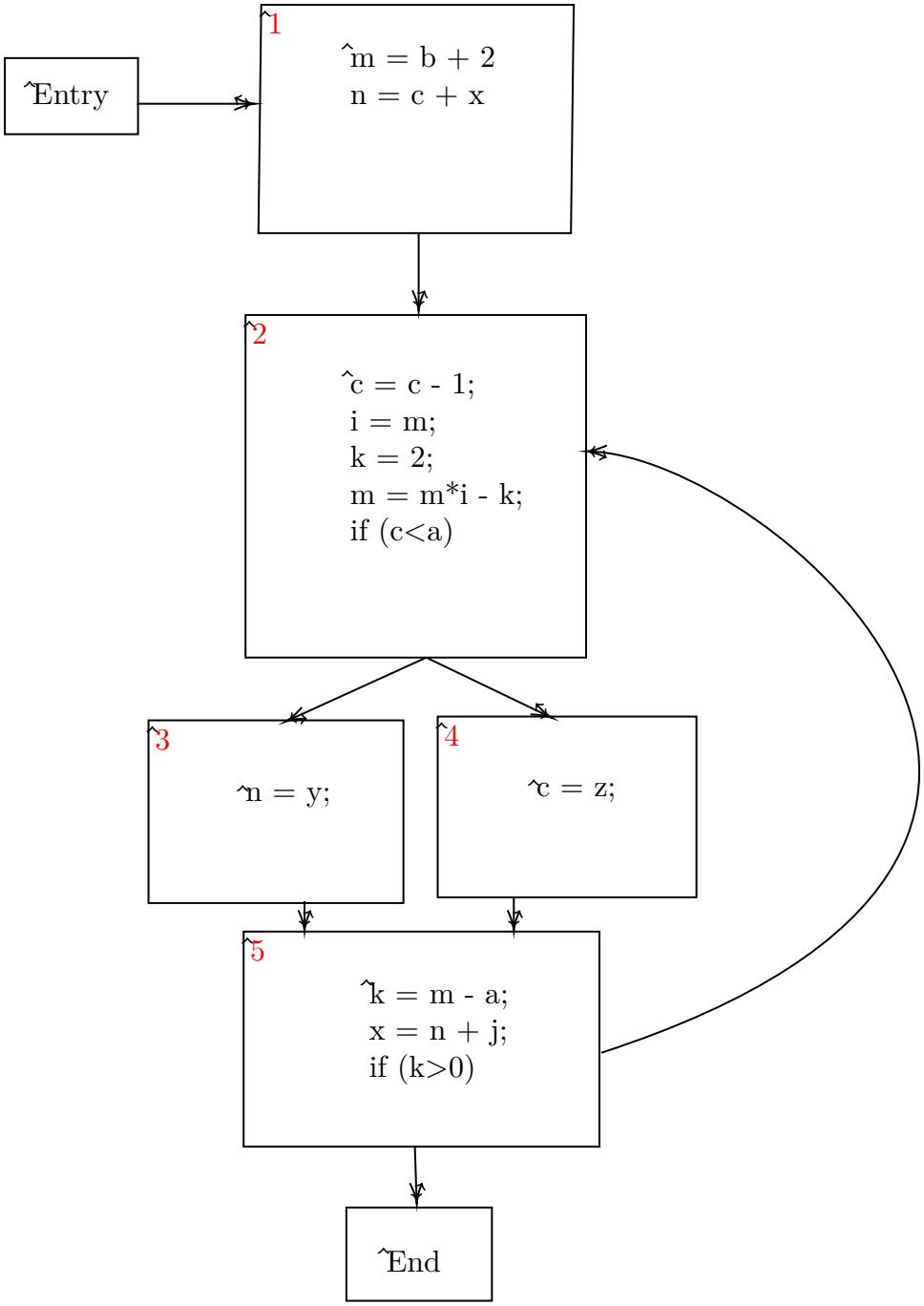
الف) در کد داده‌شده وایل به if تبدیل شده است که یا به خروج قطعه کد می‌رود و یا به بلوک ۲ می‌رود.



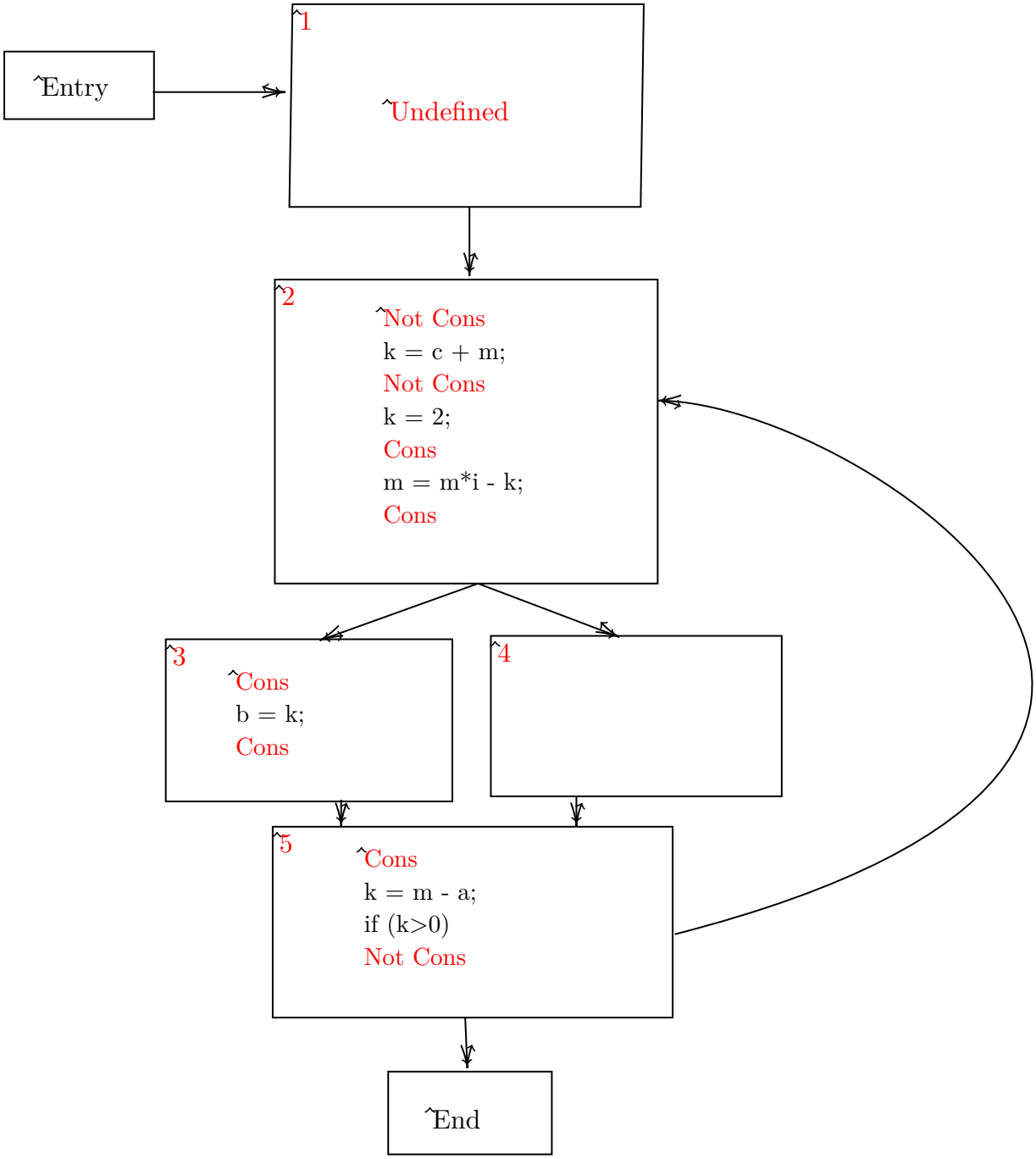
ب) تحلیل خواسته شده به صورت زیر می‌باشد: ابتدا برای بلوک ۵ مقادیر زنده در انتها را برابر با {x،m،i} قرار می‌دهیم با توجه به سه خط کد k،x حذف می‌شوند و m،a،n،j اضافه می‌شوند بنابراین مقادیر زنده در ابتدای بلوک برابر با {m،i،n،a،j} می‌باشد حال این آرایه را در انتهای بلوک ۳ و ۴ قرار می‌دهیم و با توجه به کد درون آن‌ها آرایه‌ی ابتدایی ۳ و ۴ را بدست می‌آوریم. در بلوک ۳ با توجه به خط دوم مقدار b حذف می‌شود و مقدار k اضافه می‌شود و برای خط اول هم مقدار y اضافه می‌شود و مقدار n حذف می‌شود بنابراین آرایه‌ی ابتدایی بلوک ۳ برابر {m،i،a،j،y،k} می‌باشد برای بلوک ۴ نیز به همین ترتیب پیش می‌رویم و با توجه به خط دوم مقدار l حذف می‌شود و مقادیر n،j اضافه می‌شوند و با توجه به خط اول متغیر z اضافه می‌شود و متغیر c حذف می‌شود و مقدار ابتدایی آن برابر با {m،i،n،a،j،z} می‌باشد. با توجه به دو مقدار بدست آمده برای ابتدای بلوک ۳ و ۴ مقدار انتهای بلوک ۲ برابر با اجتماع این دو مقدار می‌شود که برابر است با {m،i،n،a،j،k،y،z} . حال برای بلوک دو با توجه به قطعه کد داده شده مقدار k حذف می‌شود(خط ۲) و مقدار c قرار میگیرد همچنین مقدار i نیز حذف می‌شود(خط ۳) و مقدار ابتدایی بلوک ۲ برابر با {m،a،n،j،z،y،c} می‌باشد. حال با توجه به حلقه این مقدار را برابر با خروجی بلوک ۵ قرار می‌دهیم سپس دوباره به بالا می‌رویم و آرایه‌ها تغییراتی می‌کنند که منطق تغییرات مانند بالا می‌باشد. پس از آپدیت کردن مقدار بلوک‌های ۳،۴ و۵ نوبت به مقداردهی بلوک ۱ می‌باشد که متغیرهای زنده در انتهای آن بلوک را برابر با ابتدای بلوک دو قرار می‌دهیم و مقادیر در ابتدای بلوک برابر با {a،b،c،x،j،z،y،c} می‌باشد. در زیر شکل مربوط آمده است.



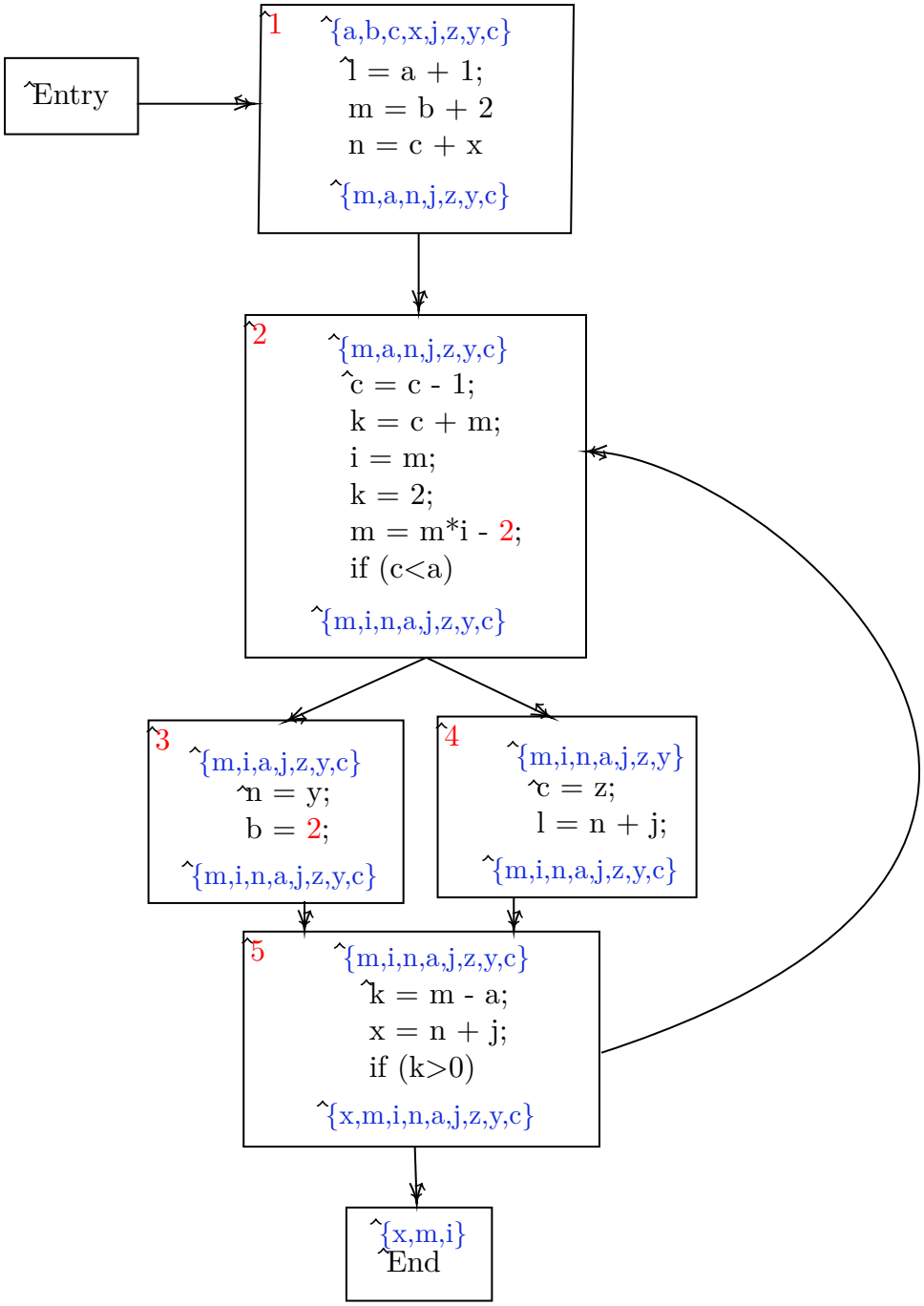
همچنین در این قسمت با توجه به وضعیت زنده بودن متغیرها می‌توان بعضیی از قسمت‌های اضافی را حذف کرد به عنوان مثال می‌توان در بلوک ۴ خط دوم را حذف کرد، همچنین می‌توان در بلوک سوم نیز خط دوم را حذف کرد. در این صورت در بلوک دوم می‌توان خط ۲ را حذف کرد و در بلوک اول نیز خط اول را می‌توانستیم حذف کنیم در نهایت کد به صورت زیر بدست می‌آید: (با استفاده از نکات مورد ج می‌توانستیم کد را بهینه تر نیز بکنیم به عنوان مثال می‌توانستیم خط ۴ بلوک یک را نیز حذف کنیم و در خط ۵ از مقدار ثابت آن استفاده کنیم)



ج) برای حل این قسمت از آنجایی که مقدار متغیر k در بعضی جاها constant است جدول را به صورتی می‌کشیم که گزاره‌های مرتبط با k در آن باشد پس جدول به صورت زیر بدست می‌آید:



بنابراین کد به صورت زیر تبدیل می‌شود از آنجایی که مقدار b برابر با constant شده می‌توان یکبار دیگر این کار را برای b انجام داد ولی از آنجایی که b جای دیگری استفاده نشده است لازم به انجام این کار نمی‌باشد. در بلوک زیر زندگی متغیرها دوباره محاسبه شد است



د) با توجه به اینکه در بلوک ۴ و ۵ از مقدار n + j استفاده شده است می‌توان تصور کرد که redundancy داریم ولی از آنجایی که در بلوک ۳ مقدار n تغییر کرده که در اجرای بلوک ۵ تاثیر می‌گذارد باعث می‌شود که مقادیر بلوک‌های ۴ و ۵ از یک جنس نباشند و نتوان با روش‌های ساده این افزونگی را رفع کرد. می‌توانستیم مقدار x + c و y+j را در بلوک یک محاسبه کنیم و از این مقادیر به گونه‌ای استفاده کنیم ولی تضمینی نیست که در این روش تعداد عملیات‌ها حتما کمتر می‌شود و از مبحث redundancy partial خارج می‌باشد.

۸. کد زیر را در نظر بگیرید.

```
a = b
c = 7 + 7 * e
d = a
a = d * d
d = 5 * a
f = c * 5 + 10
f = d - f
c = f + 1
e = c * b
print(c, b)
```

الف) گراف تداخل رجیستر ها را رسم کنید.

ب) روال اجرای الگوریتم chaitin را برروی آن توضیح دهید و کمترین تعداد رجیستر را بدست آورید.

پاسخ:

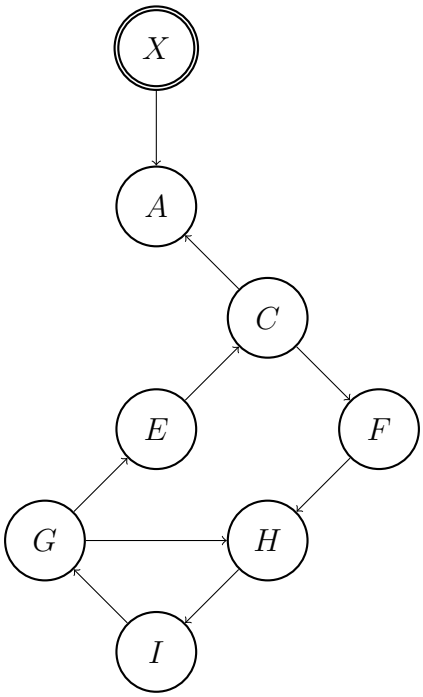
الف) CFG متناظر با کد رسم شده سپس تحلیل زنده بودن متغیر ها روی آن انجام شده و از روی نتیجه، گراف تداخل ثبات‌ها رسم شده است.

ب)حداقل تعداد ثبات مورد نیاز برای رنگ کردن گراف برابر اندازه‌ی بزرگترین گروهک گراف است که در اینجا حداقل به ۳ ثبات نیاز داریم.

روال اجرای الگوریتم chaitin :

۹. پاسخ:

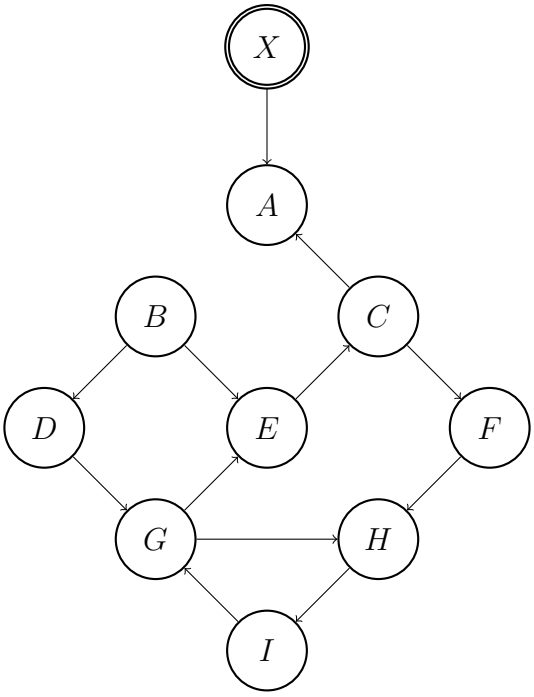
الف و ب) با حذف A->B درجه‌ی ورودی راس B برابر با صفر می‌شود پس باید حذف شود و با حذف B به همین شکل D هم حذف می‌شود و این استدلال برای هر دو بخش برقرار می‌باشد. بنابراین گراف به صورت زیر بدست می‌آید:



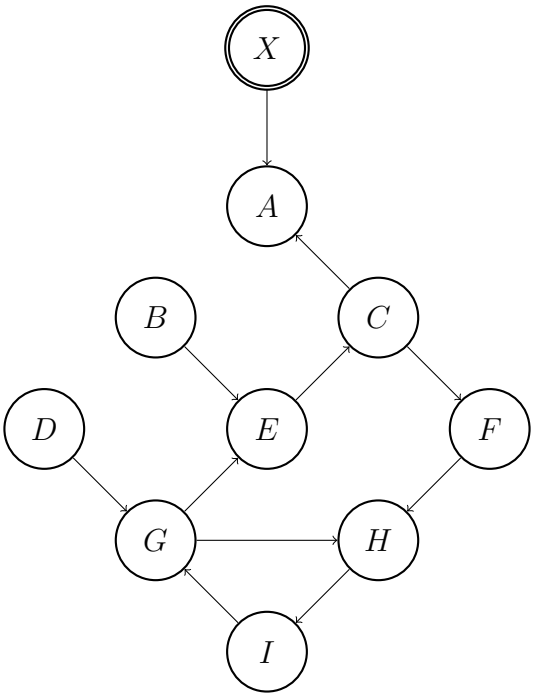
با توجه به گراف بالا هاrefcount به صورت زیر می‌باشند:

```
G : 1
I : 1
H : 2
F : 1
E : 1
C : 1
A : 2
```

همچنین اگر فرض کنیم که نودها در صورت صفر شدن refcount حذف نمی‌شوند(منظور از عدم زباله‌روبی در کوئرا چه باشد) برای قسمت اول و دوم گراف و اطلاعات به صورت زیر می‌باشند:




```
G : 2
I : 1
H : 2
F : 1
E : 2
C : 1
A : 2
D : 1
B : 0
```




```
G : 2
```

```
I : 1
H : 2
F : 1
E : 2
C : 1
A : 2
D : 0
B : 0
```

ج) با توجه به قسمت قبل متوجه می‌شویم که به هیچکدام از گره‌ها به غیر از A و X نمی‌توان رسید بنابراین اگر فرض کنیم عملیات stop-and-copy انجام می‌شود و آبجکت‌ها در درون نیمه‌ی دوم می‌باشند آنگاه بعد از زباله‌روبی در آدرس ۰ آبجکت A قرار دارد و در آدرس ۵۰ آبجکت X و بالعکس اگر در نیمه‌ی اول باشد به آدرس ۰ و ۵۰ نیمه‌ی دوم حافظه می‌رود. همچنین اگر فرض کنیم مانند hybrid از ابتدای یک حافظه شروع به پر کردن می‌کند باز هم به ترتیب در آدرس ۰ و ۵۰ آن حافظه قرار خواهند گرفت. در کل نکته‌ی اصلی همان حذف همه بجز X و A می‌باشد.