PRINCIPLES OF COMPILER DESIGN

Gate Questions and Answers

Prof.C.Naga Raju

B.Tech(CSE),M.Tech(CSE),PhD(CSE),MIEEE,MCSI,MISTE

Department of CSE

YSR Engineering College of YVU

Proddatur

INTRODUCTION

Question

GATE CS 2011

1)In a compiler, keywords of a language are recognized during

(A) parsing of the program

(B) the code generation

(C) the lexical analysis of the program

(D) dataflow analysis

OPTION C

Question

GATE CS 2011

2)The lexical analysis for a modern computer languages such as Java needs the power of which one of the following machine models is necessary and sufficient sense.

(A) Finite state automata

(B) Deterministic pushdown automata

(C) Non-Deterministic pushdown automata

(D) Turing Machine

Option A

Question

GATE-CS-2009

3) Match all items in Group 1 with correct options from those given in Group 2.

Group 1	Group 2
P. Regular expression	**1. Syntax analysis**
Q. Pushdown automata	**2. Code generation**
R. Dataflow analysis	**3. Lexical analysis**
S. Register allocation	**4. Code optimization**

(A)P-4. Q-1, R-2, S-3

(B) P-3, Q-1, R-4, S-2

(C) P-3, Q-4, R-1, S-2

(D) P-2, Q-1, R-4, S-3

OPTION B

Question

GATE CS 1998

4)Type checking is normally done during

- **(A)** Lexical analysis
- **(B)** Syntax analysis
- **(C)** Syntax directed translation
- **(D)** Code optimization

OPTION C

Question

GATE CS 2008

5) Some code optimizations are carried out on the intermediate code because

(A) they enhance the portability of the compiler to other target processors

(B) program analysis is more accurate on intermediate code than on machine code

(C) the information from dataflow analysis cannot otherwise be used for optimization

(D) the information from the front end cannot otherwise be used for optimization

OPTION A

Question

GATE CS 1997

6) A language L allows declaration of arrays whose sizes are not known during compilation. It is required to make efficient use of memory. Which of the following is true?

(A) A compiler using static memory allocation can be written for

(B) A compiler cannot be written for L, an interpreter must be used

(C) A compiler using dynamic memory allocation can be written for L

(D) None of the above

OPTION C

Question

GATE-CS-2014-(Set-3)

7) One of the purposes of using intermediate code in compilers is to

(A) make parsing and semantic analysis simpler.

(B) improve error recovery and error reporting.

(C) increase the chances of reusing the machine-independent code optimizer in other compilers.

(D) improve the register allocation.

OPTION C

Question

GATE-CS-2015 (Set 2)

8) Match the following:

List-I	List-II
A. Lexical analysis	1. Graph coloring
B. Parsing	2. DFA minimization
C. Register allocation	3. Post-order traversal
D. Expression evaluation	4. Production tree

Codes: A B C D

(A) 2 3 1 4 **(B)** 2 1 4 3 **(C)** 2 4 1 3 **(D)** 2 3 4 1

OPTION C

Question

9) In a two-pass assembler, symbol table is

(A) Generated in first pass

(B) Generated in second pass

(C) Not generated at all

(D) Generated and used only in second pass

OPTION A

Question

10) How many tokens will be generated by the scanner for the following statement ?

x = x ∗ (a + b) – 5;

(A) 12

(B) 11

(C) 10

(D) 07

- OPTION A

Question

11) Incremental-Compiler is a compiler?

(A) which is written in a language that is different from the source language

(B) compiles the whole source code to generate object code afresh

(C)compiles only those portion of source code that have been modified.

(D)that runs on one machine but produces object code for another machine

OPTION C

Question

- 12) Which phase of compiler generates stream of atoms?

(A) Syntax Analysis

(B) Lexical Analysis

(C) Code Generation

(D) Code Optimization

OPTION B

Question

14)Which data structure in a compiler is used for managing information about variables and their attributes?

A)Abstract syntax tree

B) Symbol tree

C)Semantic stack

D) Symbol table

OPTION D

Question

15) Which one of the following is NOT performed during compilation?

A)Dynamic memory allocation

B)Type checking

C)Symbol table management

D)Inline expansion

OPTION A

Question

16) Symbol table can be used for:

A) Checking type compatibility

B) Suppressing duplication of error message

C) Storage allocation

D) All of these

OPTION D

Question

17) In two-pass assembler, symbol table is

A) Generated in first pass

B) Generated in second pass

C) Not generated at all

D) Generated and used only in second pass

OPTION A

Question

18) The access time of the symbol table will be logarithmic if it is implemented by

A)Linear list

B) Search tree

C) Hash table

D) Self organization list

OPTION B

Question

19) Which one of the following is FALSE?

A) A basic block is a sequence of instructions where control enters the sequence a the beginning and exits at the end.

B)Available expression analysis can be used for common sub expression elimination.

C)Live variable analysis can be used for dead code elimination.

D) X=4*5=>x=20 is an example of common sub expression elimination

OPTION D

Question

20) One of the purposes of using intermediate code in compilers is to

A) Make parsing and semantic analysis simpler

B) Improve error recovery and error reporting

C) Increase the chances of reusing the machine independent code optimizer in other compilers

D) Improve the register allocation

OPTION C

LEXICAL ANALYSIS

Question

1)The number of tokens in the following C statement is
printf("i = %d, &i = %x", i, &i); (GATE 2000)

A.3
B.26
C.10
 D.21

OPTION C
1)Printf   2) (3)"i = %d, &i = %x“ 4) , 5) I 6), 7) & 8) I 9)) 10) ;

Question

2) In a compiler, keywords of a language are recognized during (2011)

A.parsing of the program

B.the code generation

C.the lexical analysis of the program

D.dataflow analysis

Answer is C

Question

3) The lexical analysis for a modern computer language such as Java needs the power of which one of the following machine models in a necessary and sufficient sense?

A. Finite state automata

B. Deterministic pushdown automata

C. Non-Deterministic pushdown automata

D. Turing Machine

OPTION A

Question

4) Consider the following statements:

(I) The output of a lexical analyzer is groups of characters.

(II) Total number of tokens in printf("i=%d, &i=%x", i, &i); are 14.

(III) Symbol table can be implementation by using array and hash table but not tree.

Which of the following statement(s) is/are correct?

A. Only (I)
B. Only (II) and (III)
C. All (I), (II), and (III)
D. None of these

OPTION D

Question

5) Which one of the following statements is FALSE?

A. Context-free grammar can be used to specify both lexical and syntax rules.
B. Type checking is done before parsing.
C. High-level language programs can be translated to different Intermediate Representations.
D. Arguments to a function can be passed using the program stack.

Option B

Question

7)The output of a lexical analyzer is

A. A parse tree

B. Intermediate code

C. Machine code

D. A stream of tokens

Option D

Question

8) Consider the following statements related to compiler construction :

I. Lexical Analysis is specified by context-free grammars and implemented by pushdown automata.

II. Syntax Analysis is specified by regular expressions and implemented by finite-state machine.

Which of the above statement(s) is/are correct ?

A. Only I
B. Only II
C. Both I and II
D. Neither I nor II

Option D

Question

9) Which of the following statement(s) regarding a linker software is/are true ?

I A function of a linker is to combine several object modules into a single load module.

II A function of a linker is to replace absolute references in an object module by symbolic references to locations in other modules.

A) Only I

B) Only II

C) Both I and II

D) Neither I nor II

Option (A) is correct.

Question

10) From the given data below

: a b b a a b b a a b which one of the following is not a word in the dictionary created by LZ-coding (the initial words are a, b)?

A. a b

B. b b

C. b a

D. b a a b

B and D are correct.

Question

11) The number of tokens in the following C statement is

```
    printf("i=%d, &i=%x", i&i);
```

A. 13

B. 6

C. 10

D. 9

```
printf ( "i=%d, &i=%x" , i & i ) ;
  1    2      3        4 5 6 7 8 9
```

Total nine tokens are present. So, correct option is (D)

Question

12) In compiler optimization, operator strength reduction uses mathematical identities to replace slow math operations with faster operations. Which of the following code replacements is an illustration of operator strength reduction ?

A. Replace P + P by 2 * P or Replace 3 + 4 by 7.

B. Replace P * 32 by P < < 5

C. Replace P * 0 by 0

D. Replace (P < <4) – P by P * 15

option (B) is correct.

Question

13) Debugger is a program that

A. allows to examine and modify the contents of registers
B. does not allow execution of a segment of program
C. allows to set breakpoints, execute a segment of program and display contents of register
D. All of the above

option (C) is correct.

SYNTAX ANALYSIS

Question

Consider the grammar defined by the following production rules, with

two operators ∗ and +

S --> T * P
T --> U | T * U
P --> Q + P | Q
Q --> Id
U --> Id

Which one of the following is TRUE? GATE 2014

A) + is left associative, while ∗ is right associative

B) + is right associative, while ∗ is left associative

C)Both + and ∗ are right associative

D)Both + and ∗ are left associative

option B

Explanation

- From the grammar we can find out associative by looking at grammar.
- Let us consider the 2nd production

  T -> T * U T is generating T*U recursively (left recursive)

so * is left associative.

Similarly P -> Q + P Right recursion so + is right associative.

So option B is correct.

Question

2) The grammar $A \rightarrow AA \mid (A) \mid \varepsilon$ is not suitable for predictive-parsing because the grammar is?

GATE 2005

A) ambiguous

B) left-recursive

C) right-recursive

D) A and B

OPTION D

Explanation

- Since given grammar can have infinite parse trees for string 'ε', so grammar is ambiguous, and also A → AA has left recusion. For predictive-parsing, grammar should be: Free from ambiguity
- Free from left recursion
- Free from left factoring
- Given grammar contains both ambiguity and left factoring, so it can not have predictive parser.
- We always expect first grammar free from ambiguity for parsing.

Question

3) Consider the grammar $E \rightarrow E + n \mid E \times n \mid n$ For a sentence $n + n \times n$, the handles in the right-sentential form of the reduction are ?

GATE 2005

A) n, $E + n$ and $E + n \times n$

B) n, $E + n$ and $E + E \times n$

C) n, $n + n$ and $n + n \times n$

D) n, $E + n$ and $E \times n$

OPTION D

Explanation

- $E \rightarrow E * n$ {Applying $E \rightarrow E * n$ }
- $E \rightarrow E + n * n$ {Applying $E \rightarrow E + n$ }
- $E \rightarrow n + n * n$ {Applying $E \rightarrow n$ } Hence, the handles in right sentential form is n, $E + n$ and $E \times n$.

Question

4) Which of the following is essential for converting an infix expression to the postfix from efficiently? GATE 1997

- A)An operator stack
- B) An operand stack
- C) An operand stack and an operator stack
- D) A parse tree

- option (A)

Explanation

- Operator stack is used for converting infix to postfix expression such that operators like as +, *, (,), / are pushed in stack where as operand stack is used for converting Postfix to Prefix evaluation such that operands are 7,2,1,2 etc. Hence, option (A) is correct.

Question

5. A CFG is ambiguous if

  a) It has more than one rightmost derivations

  b) It has more than one leftmost derivations

  c) No parse tree can be generated for the CFG

  d) a or b

OPTION D

- Answer: d

  Explanation: A context free grammar is ambiguous if it has more than one parse tree generated or more than one leftmost derivations or right most derivations.
- An unambiguous grammar is a context free grammar for which every valid string has a unique leftmost derivation.

Question

6. Which of the following are always unambiguous?

a) Deterministic Context free grammars

b) Non-Deterministic Regular grammars

c) Context sensitive grammar

d) None of the mentioned

OPTION A

- Answer: a

  Explanation: Deterministic CFGs are always unambiguous ,
- and are an important subclass of unambiguous CFGs;
- there are non-deterministic unambiguous CFGs, however.

Question

7. A CFG is not closed under

   a) Dot operation

   b) Union Operation

   c) Concatenation

   d) Iteration

OPTION D

- Answer: d

  Explanation: The closure property of a context free grammar does not include iteration or kleene or star operation.

Question

8. Which of the following is an example of inherent ambiguous language?

a) $\{a^n | n>1\}$

b) $\{a^n b^n c^m d^m | n,m > 0\}$

c) $\{0^n 1^n | n>0\}$

d) None of the mentioned

OPTION B

- Answer: b

  Explanation: This set is context-free, since the union of two context-free languages is always context free.

Question

9. State true or false:

Statement: R->R|T

T->ε

is an ambiguous grammar

a) true

b) false

OPTION A

- Answer: a

  Explanation: The production can be either itself or an empty string.
- Thus the empty string has more than one leftmost derivations, depending on how many times R->R is being used.

Question

1) Which of the following be sufficient to convert an arbitrary CFG to an LL(1) grammar?

A) Removing left recursion alone

B) Factoring the grammar alone

C) Removing left recursion and factoring the grammar

D) None of these

Option D

Explanation

- Removing left recursion and factoring the grammar do not be sufficient to convert an arbitrary CFG to LL(1) grammar

Question

2) Which one of the following is a top-down parser?

A. Recursive descent parser.
B. Operator precedence parser.
C. An LR(k) parser.
D. An LALR(k) parser

Option A

Explanation

- Recursive Descent parsing is LL(1) parsing which is top down parsing.

Question

3)Which of the following derivations does a top-down parser use while parsing an input string? The input is assumed to be scanned in left to right order (GATE CS 2000).

A. Leftmost derivation
B. Leftmost derivation traced out in reverse
C. Rightmost derivation
D. Rightmost derivation traced out in reverse

Option A

Answer (a)

- In top down parsing, we just start with the start symbol and compare the right side of the different productions against the first piece of input to see which of the productions should be used.
- A top down parser is called LL parser because it parses the input from **L**eft to right, and constructs a **L**eftmost derivation of the sentence.

Question

4.For the grammar below, a partial LL(1) parsing table is also presented along with the grammar. Entries that need to be filled are indicated as E1, E2, and E3. |epsilon is the empty string, $ indicates end of input, and, | separates alternate right hand sides of productions.

$S \rightarrow a\,A\,b\,B \mid b\,A\,a\,B \mid \varepsilon$
$A \rightarrow S$
$B \rightarrow S$

	a	b	$
S	**E1**	**E2**	$S \rightarrow \varepsilon$
A	$A \rightarrow S$	$A \rightarrow S$	error
B	$B \rightarrow S$	$B \rightarrow S$	**E3**

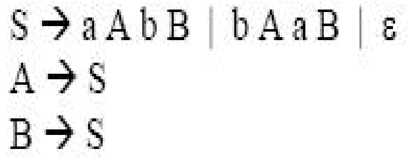
(A) FIRST(A) = {a, b, ε} = FIRST(B)
FOLLOW(A) = {a, b}
FOLLOW(B) = {a, b, $}

(B) FIRST(A) = {a, b, $}
FIRST(B) = {a, b, ε}
FOLLOW(A) = {a, b}
FOLLOW(B) = {$}

(C) FIRST(A) = {a, b, ε} = FIRST(B)
FOLLOW(A) = {a, b}
FOLLOW(B) = ∅

(D) FIRST(A) = {a, b} = FIRST(B)
FOLLOW(A) = {a, b}
FOLLOW(B) = {a, b}

Answer: (A)



$S \rightarrow a A b B \mid b A a B \mid \varepsilon$

$A \rightarrow S$

$B \rightarrow S$

Now in the above question,

FIRST(S) = { a, b, epsilon}

FIRST(A) = FIRST(S) = { a, b, epsilon}

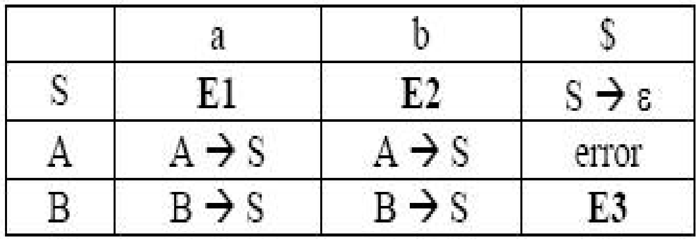
FIRST(B) = FIRST(S) = { a, b, epsilon}

FOLLOW (A) = { b , a }

FOLLOW (S) = { $ } U FOLLOW (A) = { b , a , $ }

FOLLOW (B) = FOLLOW (S) = { b ,a , $ }

epsilon corresponds to empty string.



	a	b	$
S	**E1**	**E2**	$S \rightarrow \varepsilon$
A	$A \rightarrow S$	$A \rightarrow S$	error
B	$B \rightarrow S$	$B \rightarrow S$	**E3**

Question

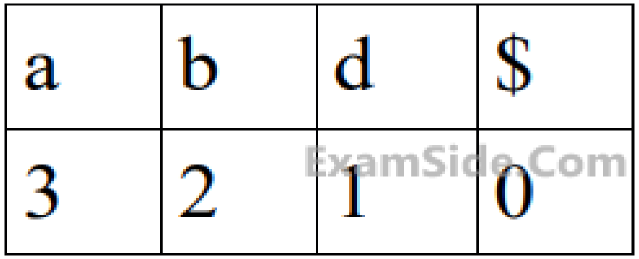
5.Consider the grammar given below:

S → Aa

A → BD

B → b | ε

D → d | ε



a	b	d	$
3	2	1	0

Let a, b, d, and $ be indexed as follows:

Compute the FOLLOW set of the non-terminal B and write the index values for the symbols in the FOLLOW set in the descending order. (For example, if the FOLLOW set is {a, b, d, $}, then the answer should be 3210)

Your Input _________(GATE CSE 2019)

- Follow(B) = First(D) ∪ Follow(A)

  [When D is ε then Follow(B) = Follow(A)]

  ∴ Follow(B) = {d} ∪ {a} = {a, d}

  As a = 3 and d = 1 then Descending order = 31

Question

6.The grammar $A \rightarrow AA|(A)|\varepsilon A \rightarrow AA|(A)|\varepsilon$

is not suitable for predictive-parsing because the grammar is

A. ambiguous

B. left-recursive

C. right-recursive

D. Both A and B

Option D

SYNTAX ANALYSIS
BOTTOM UP PARSERS

Question

Consider the grammar

$E \rightarrow E + n \mid E \times n \mid n$

For a sentence $n + n \times n$, the handles in the right-sentential form of the reductions are ? (GATE 2005)

A. n, $E + n$ and $E + n \times n$

B. n, $E + n$ and $E + E \times n$

C. n, $n + n$ and $n + n \times n$

D. n, $E + n$ and $E \times n$

Option D

Explanation

- E → E * n {Applying E → E * n }
- E→ E + n * n {Applying E → E + n }
- E→ n + n * n {Applying E → n } Hence, the handles in right sentential form is n, E + n and E × n.
- Hence **Option D is the right choice**

Question

A bottom-up parser generates:

A. Left-most derivation in reverse
B. Right-most derivation in reverse
C. Left-most derivation
D. Right-most derivation

Option (B)

Explanation

- A bottom-up parser generates right-most derivation in reverse
- **Option (B) is correct.**

Question

Consider the following statements related to compiler construction :

I. Lexical Analysis is specified by context-free grammars and implemented by pushdown automata.

II. Syntax Analysis is specified by regular expressions and implemented by finite-state machine.

Which of the above statement(s) is/are correct ?

A. Only I

B. Only II

C. Both I and II

D. Neither I nor II

option (D)

Explanation

- Both statements are wrong for detailed information on lexical analysis and syntax analysis
- **option (D) is correct.**

Question

Which of these is true about LR parsing?

A. Is most general non-backtracking shift-reduce parsing
B. It is still efficient
C. Both a and b
D. None of the mentioned

option (C)

Explanation

- LR parsers are a type of bottom-up parsers that efficiently handle deterministic context-free languages in guaranteed linear time.
- **option (C) is correct.**

Question

Which of the following is incorrect for the actions of A LR-Parser
I) shift s

ii) reduce A->ß

iii) Accept

iv) reject?
Only I)

A. I) and ii)

B. I), ii) and iii)

C. I), ii) , iii) and iv)

D. none

Option C

Explanation

- Only reject out of the following is a correct LR parser action
- **Option C is correct**

Question

If a state does not know whether it will make a shift operation or reduction for a terminal is called

A. Shift/reduce conflict
B. Reduce /shift conflict
C. Shift conflict
D. Reduce conflict

Option A

Explanation

- As the name suggests that the conflict is between shift and reduce hence it is called shift reduce conflict
- **Option A is correct**

Question

When there is a reduce/reduce conflict?

A. If a state does not know whether it will make a shift operation using the production rule i or j for a terminal.
B. If a state does not know whether it will make a shift or reduction operation using the production rule i or j for a terminal.
C. If a state does not know whether it will make a reduction operation using the production rule i or j for a terminal.
D. None of the mentioned

Option C

Explanation

- It occurs when If a state does not know whether it will make a reduction operation using the production rule i or j for a terminal.
- **Option C is correct**

Question

- Consider the following two statements: P: Every regular grammar is LL(1) Q: Every regular set has a LR(1) grammar Which of the following is TRUE? (GATE 2007)

A. Both P and Q are true
B. P is true and Q is false
C. P is false and Q is true
D. Both P and Q are false

option C

Explanation

- A regular grammar can also be ambiguous also For example, consider the following grammar, S → aA/a A → aA/ε In above grammar, string 'a' has two leftmost derivations.
- (1) S → aA (2) S → a S->a (using A->ε) And LL(1) parses only unambiguous grammar, so statement P is False.
- Statement Q is true is for every regular set, we can have a regular grammar which is unambiguous so it can be parse by LR parser.
- So **option C** is correct choice

Question

A canonical set of items is given below

$S \rightarrow L. > R$

$Q \rightarrow R.$

On input symbol < the set has? (GATE 2014)

A. a shift-reduce conflict and a reduce-reduce conflict.

B. a shift-reduce conflict but not a reduce-reduce conflict.

C. a reduce-reduce conflict but not a shift-reduce conflict.

D. neither a shift-reduce nor a reduce-reduce conflict

option D

Explanation

- The question is asked with respect to the symbol ' < ' which is **not present** in the given canonical set of items.
- Hence it is neither a shift-reduce conflict nor a reduce-reduce conflict on symbol '<‘.
- So **option D** is correct choice
- But if the question would have asked with respect to the symbol ' > ' then it would have been a shift-reduce conflict.

Question

Which of the following is true?

A. Canonical LR parser is LR (1) parser with single look ahead terminal

B. All LR(K) parsers with K > 1 can be transformed into LR(1) parsers.

C. Both (A) and (B)

D. None of the above

Option (C)

Explanation

- Canonical LR parser is LR (1) parser with single look ahead terminal. All LR(K) parsers with K > 1 can be transformed into LR(1) parsers.
- **Option (C) is correct.**

Question

The construction of the canonical collection of the sets of LR (1) items are similar to the construction of the canonical collection of the sets of LR (0) items. Which is an exception?

A. Closure and goto operations work a little bit different
B. Closure and goto operations work similarly
C. Closure and additive operations work a little bit different
D. Closure and associatively operations work a little bit different

Option (A)

Explanation

- Closure and goto do work differently in case of LR (0) and LR (1)
- **Option (A) is correct.**

Question

When ß (in the LR(1) item A -> ß.a,a) is not empty, the look-head

A. Will be affecting.
B. Does not have any affect.
C. Shift will take place.
D. Reduction will take place.

Option (B)

Explanation

- There is no terminal before the non terminal beta
- **Option (B) is correct.**

Question

When ß is empty (A -> ß.,a), the reduction by A-> a is done

A. If next symbol is a terminal
B. Only If the next input symbol is a
C. Only If the next input symbol is A
D. Only if the next input symbol is a

Option (D)

Explanation

- The next token is considered in this case it’s a
- **Option (D) is correct.**

Question

- Consider the following two statements:
- P: Every regular grammar is LL(1)
- Q: Every regular set has a LR(1) grammar
- Which of the following is TRUE? (GATE 2007)

A. Both P and Q are true
B. P is true and Q is false
C. P is false and Q is true
D. Both P and Q are false

option C

- A regular grammar can also be ambiguous
- For example, consider the following grammar,
- S → aA/a
- A → aA/ε In above grammar, string 'a' has two leftmost derivations.
- (1) S → aA
- (2) S → a
- S->a (using A->ε)
- And LL(1) parses only unambiguous grammar, so statement P is False.
- Statement Q is true is for every regular set, we can have a regular grammar which is unambiguous so it can be parse by LR parser.
- So **option C** is correct choice

Question

A canonical set of items is given below

$S \rightarrow L. > R$

$Q \rightarrow R.$

On input symbol < the set has? (GATE 2014)

A. a shift-reduce conflict and a reduce-reduce conflict.
B. a shift-reduce conflict but not a reduce-reduce conflict.
C. a reduce-reduce conflict but not a shift-reduce conflict.
D. neither a shift-reduce nor a reduce-reduce conflict

Option D

Explanation

- The question is asked with respect to the symbol ' < ' which is **not present** in the given canonical set of items.
- Hence it is neither a shift-reduce conflict nor a reduce-reduce conflict on symbol '<‘.
- So **option D** is correct choice
- But if the question would have asked with respect to the symbol ' > ' then it would have been a shift-reduce conflict.

Question

Which of the following is true?

A. Canonical LR parser is LR (1) parser with single look ahead terminal
B. All LR(K) parsers with K > 1 can be transformed into LR(1) parsers.
C. Both (A) and (B)
D. None of the above

Option (C)

Explanation

- Canonical LR parser is LR (1) parser with single look ahead terminal. All LR(K) parsers with K > 1 can be transformed into LR(1) parsers.
- **Option (C) is correct.**

Question

The construction of the canonical collection of the sets of LR (1) items are similar to the construction of the canonical collection of the sets of LR (0) items. Which is an exception?

A. Closure and goto operations work a little bit different
B. Closure and goto operations work similarly
C. Closure and additive operations work a little bit different
D. Closure and associatively operations work a little bit different

- **Option (A)**

Explanation

- Closure and goto do work differently in case of LR (0) and LR (1)
- **Option (A) is correct.**

Question

When ß (in the LR(1) item A -> ß.a,a) is not empty, the look-head

A. Will be affecting.
B. Does not have any affect.
C. Shift will take place.
D. Reduction will take place.

Option (B)

Explanation

- There is no terminal before the non terminal beta
- **Option (B) is correct.**

Question

When ß is empty (A -> ß.,a), the reduction by A-> a is done

A. If next symbol is a terminal
B. Only If the next input symbol is not a
C. Only If the next input symbol is A
D. Only if the next input symbol is a

Option (D)

Explanation

- The next token is considered in this case it’s a
- **Option (D) is correct.**

Question

Which one from the following is false?

A. LALR parser is Bottom - Up parser
B. A parsing algorithm which performs a left to right scanning and a right most deviation is RL (1)
C. LR parser is Bottom - Up parser.
D. In LL(1), the 1 indicates that there is a one - symbol look - ahead.

option (B)

Explanation

- LALR parser is Bottom - Up parser.True
- A parsing algorithm which performs a left to right scanning and a right most deviation is RL (1).False
- LR parser is Bottom - Up parser. True
- In LL(1), the 1 indicates that there is a one - symbol look - ahead.True
- So, **option (B) is correct**.

Question

Which of the following is the most powerful parsing method?

A. LL(1)
B. Canonical LR
C. SLR
D. LALR

Option B

Question

Which of the following statement is true?

A. SLR parser is more powerful than LALR.
B. LALR parser is more powerful than Canonical LR parser.
C. Canonical LR parser is more powerful than LALR parser.
D. The parsers SLR, Canonical LR, and LALR have the same power

Option C

Question

Which of the following statements is false?

A. An unambiguous grammar has same leftmost and rightmost derivation
B. An LL(1) parser is a top-down parser
C. LALR is more powerful than SLR
D. An ambiguous grammar can never be LR(k) for any k

Option A

Explanation

- **Option A is correct**
- A grammar is ambiguous if there exists a string s such that the grammar has more than one leftmost derivations for s. We could also come up with more than one rightmost derivations for a string to prove the above proposition, but not both of right and leftmost. An unambiguous grammar can have different rightmost and leftmost derivations.

Question

What is the similarity between LR, LALR and SLR?

A. Use same algorithm, but different parsing table.
B. Same parsing table, but different algorithm.
C. Their Parsing tables and algorithm are similar but uses top down approach.
D. Both Parsing tables and algorithm are different.

Option A

Explanation

- The common grounds of these 3 parser is the algorithm but parsing table is different
- **Option A is correct**

SEMANTIC ANALYSIS

Question

1.Which of the following comment about peephole optimization is true?

A)It is applied to a small part of the code and applied repeatedly

B)It can be used to optimize intermediate code

C)It can be applied to a portion of the code that is not contiguous

D)It is applied in the symbol table to optimize the memory requirements.

OPTION A

- **Explanation:**
- It is applied to a small part of the code and applied repeatedly

Question

2)Which of the following class of statement usually produces no executable code when compiled?

A)Declaration

B)Assignment statements

C)Input and output statements

D)Structural statements

OPTION A

Question 2 Explanation:

Declaration

Question

3)Which of the following class of statement usually produces no executable code when compiled?

A)Declaration

B)Assignment statements

C)Input and output statements

D)Structural statements

OPTION A

Question

4)Substitution of values for names (whose values are constants) is done in

A)Local optimization

B)Loop optimization

C)Constant folding

D)Strength reduction

OPTION C

Question

5)In compiler terminology reduction in strength means

A)Replacing run time computation by compile time computation

B)Removing loop invariant computation

C)Removing common subexpressions

D)Replacing a costly operation by a relatively cheaper one

OPTION D

Question

6)Which of the following statements about peephole optimization is False?

A)It is applied to a small part of the code

B)It can be used to optimize intermediate code

C)To get the best out of this, it has to be applied repeatedly

D)It can be applied to the portion of the code that is not contiguous

OPTION D

Question

7)The graph that shows basic blocks and their successor relationship is called:

A)DAG

B)Control graph

C)Flow graph

D)Hamiltonian graph

- OPTION C

Question

8)In compiler optimization, operator strength reduction uses mathematical identities to replace slow math operations with faster operations. Which of the following code replacements is an illustration of operator strength reduction ?

A)Replace P + P by 2 * P or Replace 3 + 4 by 7.

B)Replace P * 32 by P << 5

C)Replace P * 0 by 0

D)Replace (P << 4) – P by P * 15

OPTION B

Question

9)In _______, the bodies of the two loops are merged together to form a single loop provided that they do not make any references to each other.

A)Loop unrolling

B)Strength reduction

C)Loop concatenation

D)Loop jamming

OPTION D

Question

10)Loop unrolling is a code optimization technique:

A)That avoids tests at every iteration of the loop.

B)That improves performance by decreasing the number of instructions in a basic block.

C)That exchanges inner loops with outer loops

D)That reorders operations to allow multiple computations to happen in parallel

OPTION A

Question

12)Peer-hole optimization is a form of :

A)Loop optimization

B)Local optimization

C)Constant folding

D)Data flow analysis

OPTION C

Question

13)Dead-code elimination in machine code optimization refers to :

A) Removal of all labels.

B) Removal of values that never get used.

C) Removal of function which are not involved.

D) Removal of a module after its use.

OPTION B

Question

14)In the context of compiler design, “reduction in strength” refers to :

A)Code optimization obtained by the use of cheaper machine instructions

B)Reduction in accuracy of the output

C)Reduction in the range of values of input variables

D)Reduction in efficiency of the program

OPTION A

Question

15)Some code optimizations are carried out on the intermediate code because

A)They enhance the portability of the compiler to other target processors

B)Program analysis is more accurate on intermediate code than on machine code

C)The information from dataflow analysis cannot otherwise be used for optimization

D)The information from the front end cannot otherwise be used for optimization

OPTION A and B

QUESTION NO 15 EXPLANATION:

Option (B) is also true. But the main purpose of doing some code-optimization on intermediate code generation is to enhance the portability of the compiler to target processors. So Option A) is more suitable here. Intermediate code is machine/architecture independent code. So a compiler can optimize it without worrying about the architecture on which the code is going to execute (it may be the same or the other). So that kind of compiler can be used by multiple different architectures. In contrast to that, suppose code optimization is done on target code, which is machine/architecture dependent, then the compiler has be specific about the optimizations on that kind of code. In this case the compiler can't be used by multiple different architectures, because the target code produced on different architectures would be different. Hence portability reduces here.

Question

17)Which one of the following is FALSE?

A)A basic block is a sequence of instructions where control enters the sequence at the beginning and exits at the end.

B)Available expression analysis can be used for common sub expression elimination.

C)Live variable analysis can be used for dead code elimination.

D)$x = 4 * 5 \Rightarrow x = 20$ is an example of common subexpression elimination.

OPTION D

Question

18)One of the purposes of using intermediate code in compilers is to

A)make parsing and semantic analysis simpler.

B)improve error recovery and error reporting.

C)increase the chances of reusing the machine-independent code optimizer in other compilers.

D)improve the register allocation.

option (C)

- **Question 18 Explanation:**
- After semantic Analysis, the code is converted into intermediate code which is platform(OS + hardware) independent, the advantage of converting into intermediate code is to improve the performance of code generation and to increase the chances of reusing the machine-independent code optimizer in other compilers. So, option (C) is correct.

Question

19)Consider the following C code segment.for (i = 0, i<n; i++)

- {
- for (j=0; j<n; j++) {
- if (i%2) {
- x += (4*j + 5*i);
- y += (7 + 4*j); } }}

Which one of the following is false?

A)The code contains loop invariant computation

B)There is scope of common sub-expression elimination in this code

C)There is scope of strength reduction in this code

D)There is scope of dead code elimination in this code

OPTION D

- **Question 19 Explanation:**

Question asks about **false** statement 4*j is common sub expression elimination so B is true. 5*i can be moved out of inner loop so can be i%2. Means, A is true as we have loop invariant computation. Next, 4*j as well as 5*i can be replaced with a = - 4; before j loop then a = a + 4; where 4*j is computed, likewise for 5*i. C is true as there is scope of strength reduction. By choice elimination, we have D.

Question

20)Consider the grammar rule E → E1 - E2 for arithmetic expressions. The code generated is targeted to a CPU having a single user register. The subtraction operation requires the first operand to be in the register. If E1 and E2 do not have any common sub expression, in order to get the shortest possible code

A)E1 should be evaluated first

B)E2 should be evaluated first

C)Evaluation of E1 and E2 should necessarily be interleaved

D)Order of evaluation of E1 and E2 is of no consequence

OPTION B

- **Question 20 Explanation:**
- E -> E1 - E2 Given that E1 and E2 don't share any sub expression, most optimized usage of single user register for evaluation of this production rule would come only when E2 is evaluated before E1. This is because when we will have E1 evaluated in the register, E2 would have been already computed and stored at some memory location. Hence we could just use subtraction operation to take the user register as first operand, i.e. E1 and E2 value from its memory location referenced using some index register or some other form according to the instruction. Hence correct answer should be (B) E2 should be evaluated first.

Question

21)A grammar that is both left and right recursive for a non-terminal is

A)Ambiguous

B)Unambiguous

C)Information is not sufficient to decide whether it is ambiguous or Unambiguous.

D)None of the above

OPTION C

Question

The identification of common sub-expression and replacement of runtime computations by compile-time computations is:

A.Local optimisation
B.Constant folding
C.Loop Optimisation
D.Data flow analysis

OPTION A

Question

Code optimisation is responsibility of:

A.Application programmer
B.Syatem programmer
C.Operating System
D.All of the above

OPTION D

Question

In compiler design ‘reducing the strength’refers to

A. reducing the range refers to values of input variables.
B.code optimisation using cheaper machine instruction.
C.reducing efficiency of program
D.None of the above

OPTION B

Question

Dead-code elimination in machine code optimisation refers to :

A.removal of all labels .
B.removal of values that never get used.
C.removal of function which are not involved.
D.removal of a module after its use.

OPTION B

Question

In the context of compiler design,”reduction in strength”refers to:

A. Code optimisation obtained by the use of cheaper machine instruction.
B. reduction in accuracy of the output
C. reduction in the range of values of input variables.
D. reduction in efficiency of the program

OPTION A

Question

Incompatable types work with the _______________

A. Syntax tree

B.semantic analyzer

C.Code optimizer

D.Lexical analyzer

Answer is B

Question

Consider the basic block given below.

A = b + c

c = a + d

d = b + c

e = d - b

a = e + b

The minimum number of nodes and edges present in the DAG representation of the above basic block respectively are

(A) 6 and 6

(B) 8 and 10

(C) 9 and 12

(D) 4 and 4

Answer : (A) 6 and 6

Question

Consider the following Syntax Directed Translation Scheme (SDTS), with non-terminals {S, A} and terminals {**a,b**}.

S→aA{print1} S→a{print2} S→Sb{print3}

Using the above SDTS, the output printed by a bottom-up parser, for the input **aab** is:

(A) 1 3 2

(B) 2 2 3

(C) 2 3 1

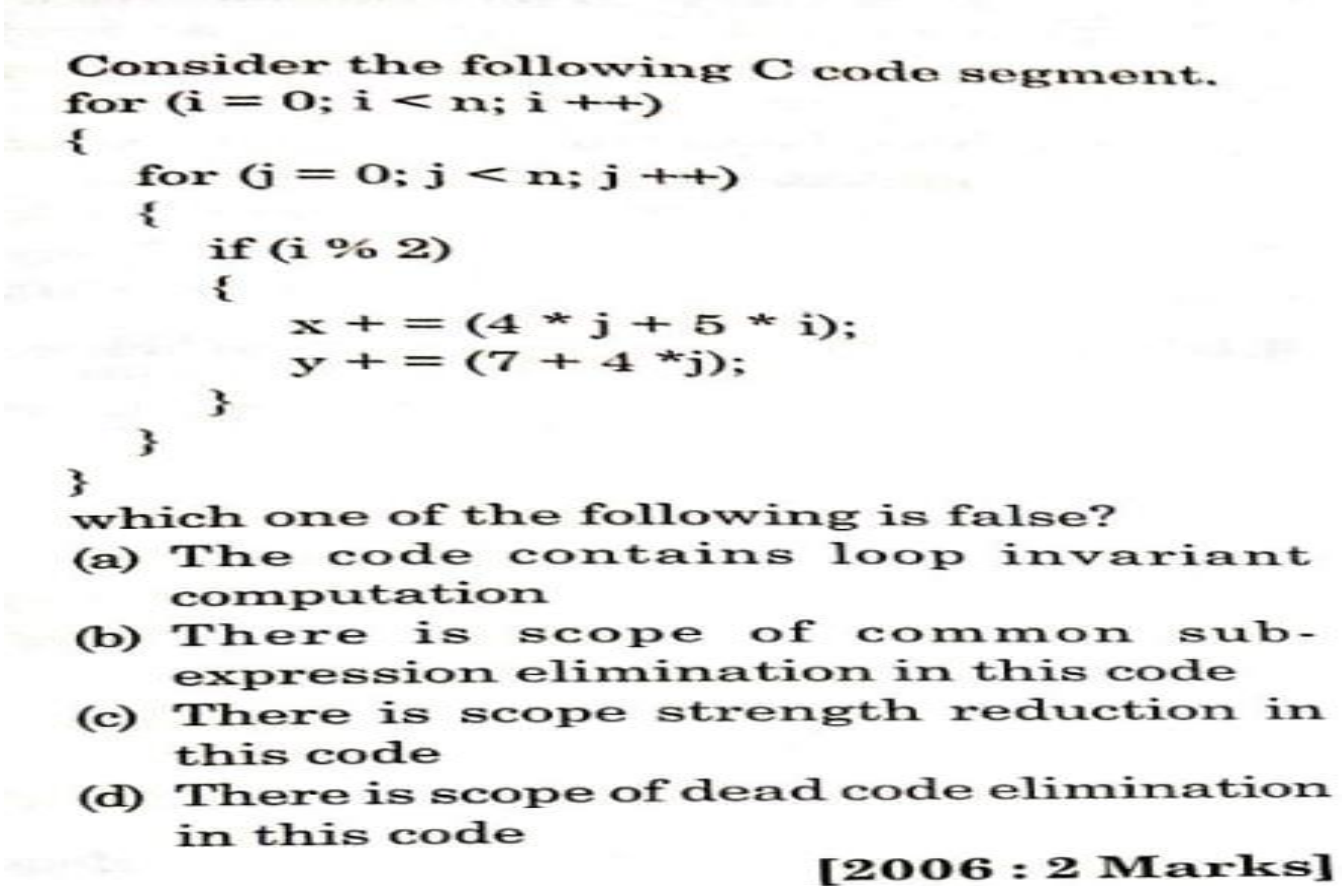
(D) syntax error

Answer : (C)

- **Answer :** (C) 2 3 1
- SOLUTION: Input is 'aab'
- So tree for given input.
- So output will be 2, 3 and 1 because printed order will be 1, 2, 3.
- So output: 2 3 1

INTERMEDIATE CODE GENERATION

Question



Consider the following C code segment.

```
for (i = 0; i < n; i ++)
{
    for (j = 0; j < n; j ++)
    {
        if (i % 2)
        {
            x + = (4 * j + 5 * i);
            y + = (7 + 4 *j);
        }
    }
}
```

which one of the following is false?

(a) The code contains loop invariant computation

(b) There is scope of common sub-expression elimination in this code

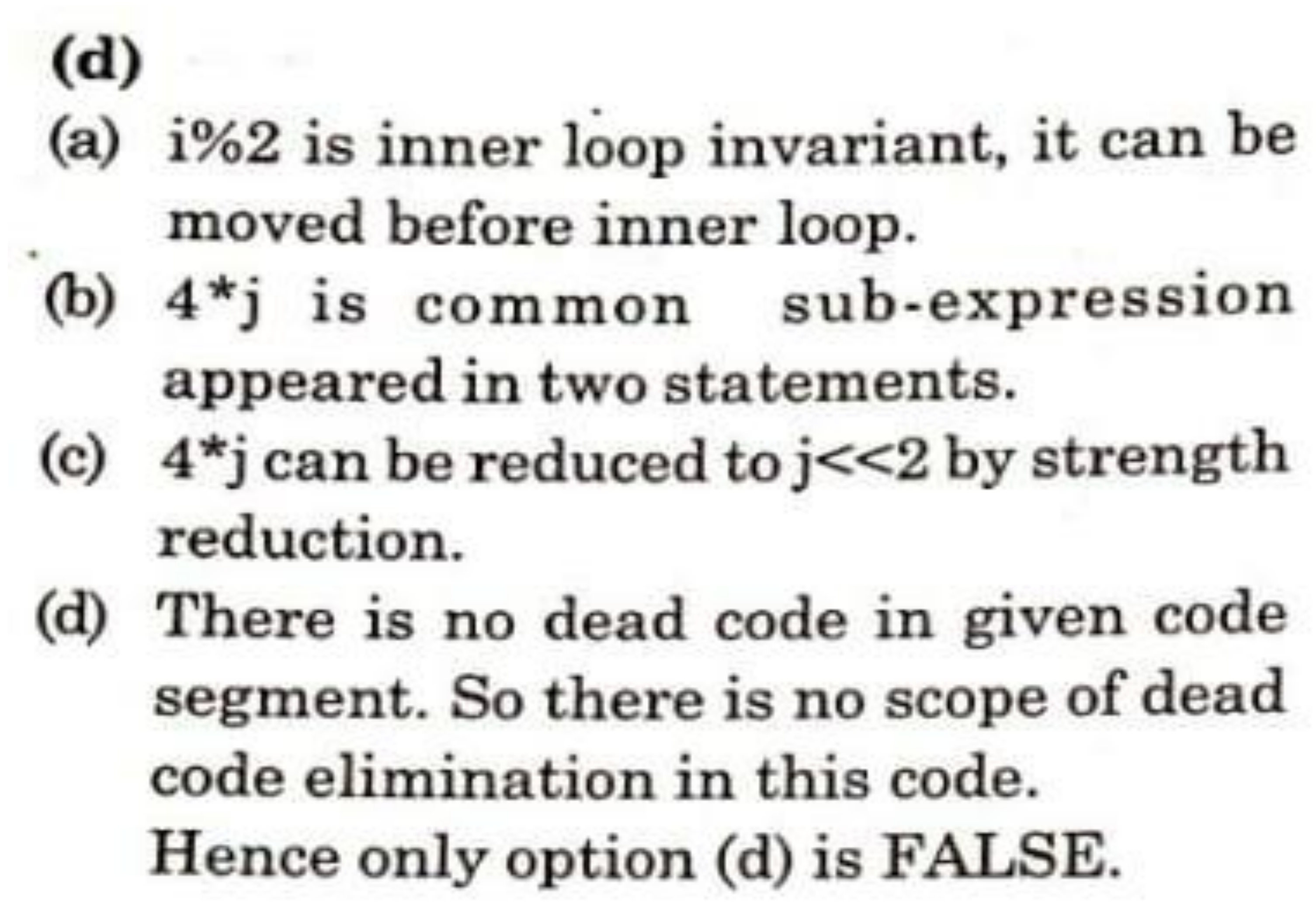
(c) There is scope strength reduction in this code

(d) There is scope of dead code elimination in this code

[2006 : 2 Marks]

OPTION D

Explanation



(d)

(a) i%2 is inner loop invariant, it can be moved before inner loop.

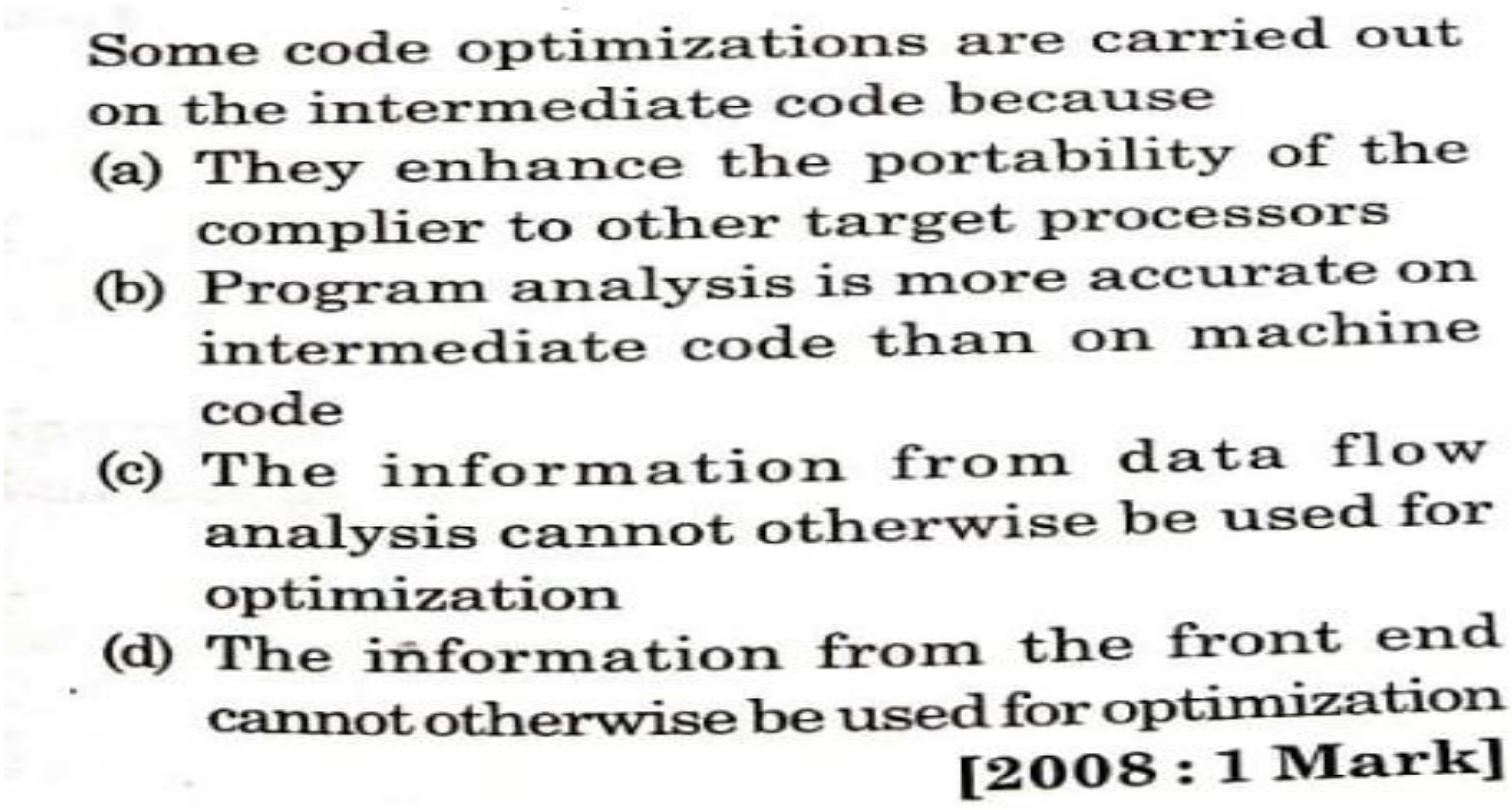
(b) 4*j is common sub-expression appeared in two statements.

(c) 4*j can be reduced to j<<2 by strength reduction.

(d) There is no dead code in given code segment. So there is no scope of dead code elimination in this code.

Hence only option (d) is FALSE.

Question



Some code optimizations are carried out on the intermediate code because

(a) They enhance the portability of the complier to other target processors

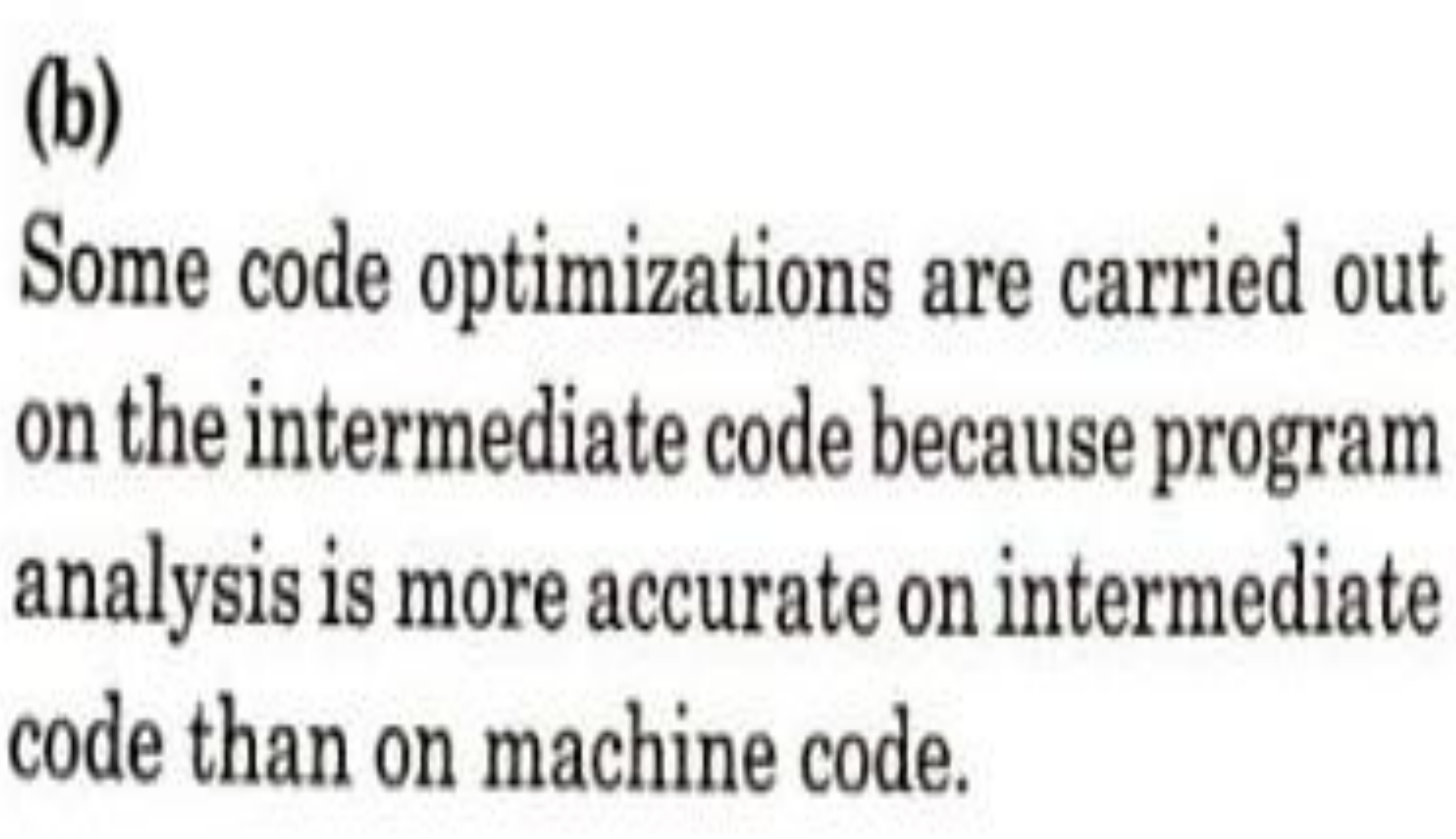
(b) Program analysis is more accurate on intermediate code than on machine code

(c) The information from data flow analysis cannot otherwise be used for optimization

(d) The information from the front end cannot otherwise be used for optimization

[2008 : 1 Mark]

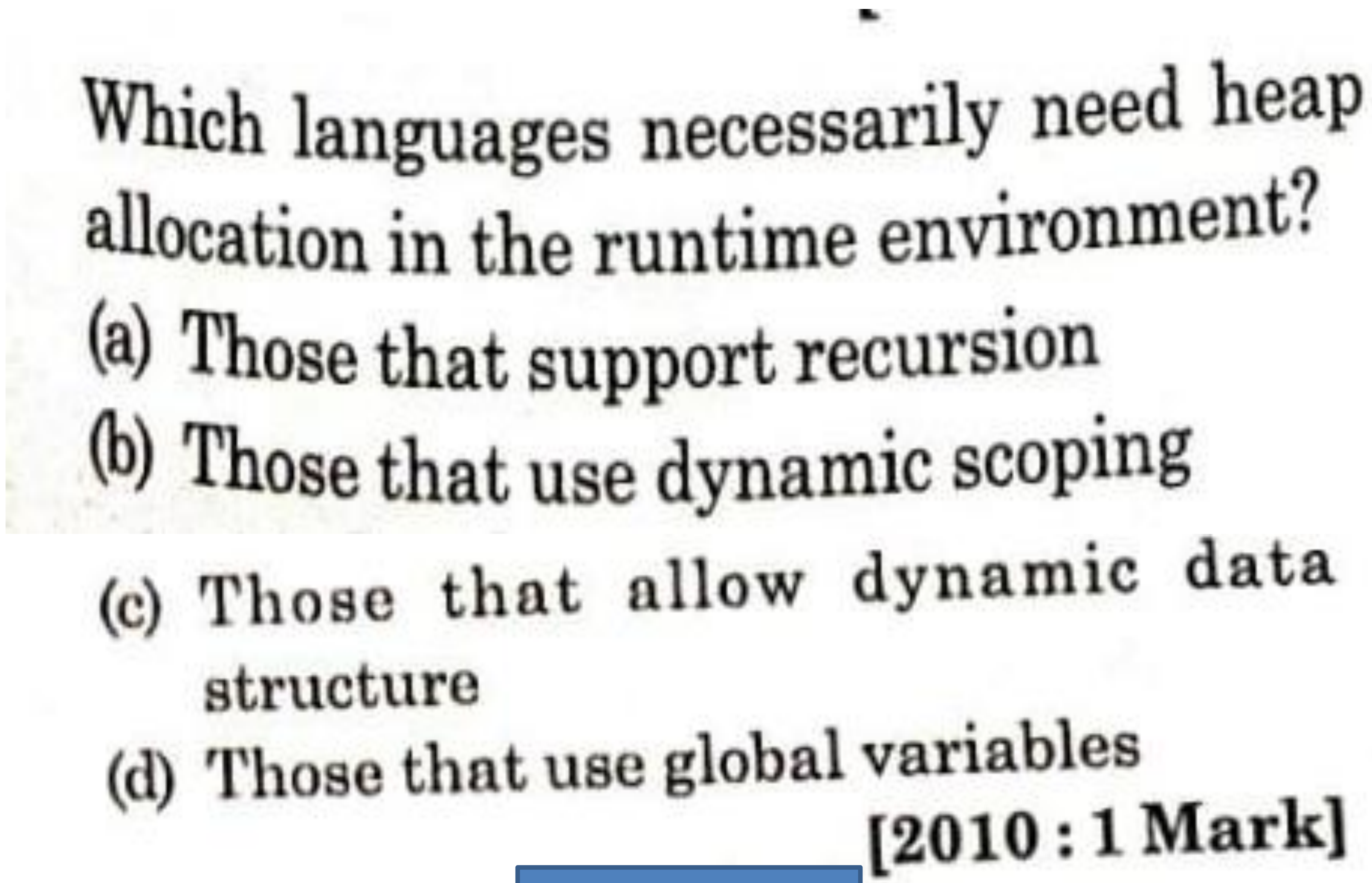
OPTION B



(b)

Some code optimizations are carried out on the intermediate code because program analysis is more accurate on intermediate code than on machine code.

Question



Which languages necessarily need heap allocation in the runtime environment?

(a) Those that support recursion

(b) Those that use dynamic scoping

(c) Those that allow dynamic data structure

(d) Those that use global variables

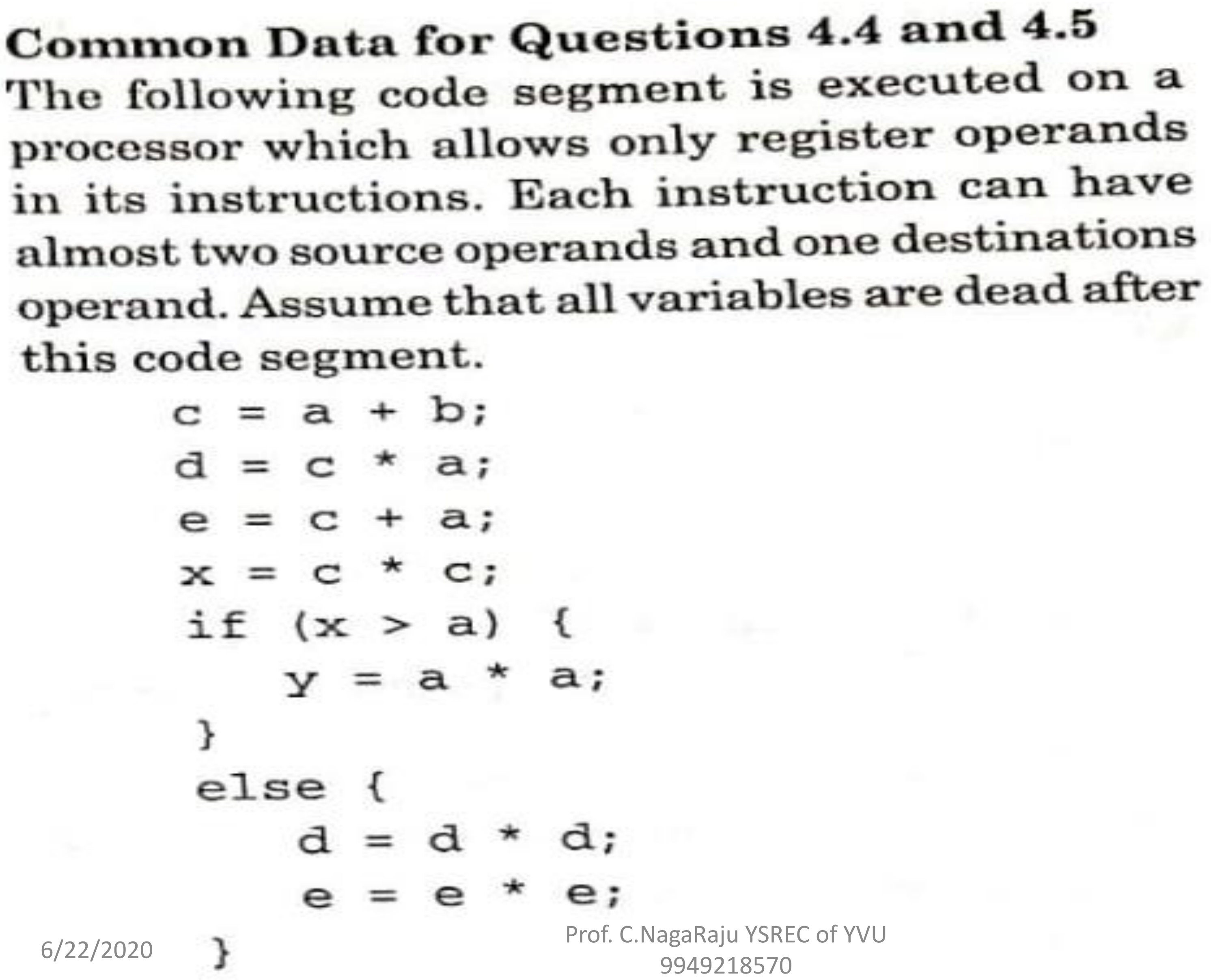
[2010 : 1 Mark]

OPTION C

(c)

Runtime environment means we deal with dynamic memory allocation and Heap is a dynamic data structure.
So it is clear that those languages that allow dynamic data structure necessarily need heap allocation in the runtime environment.

Question

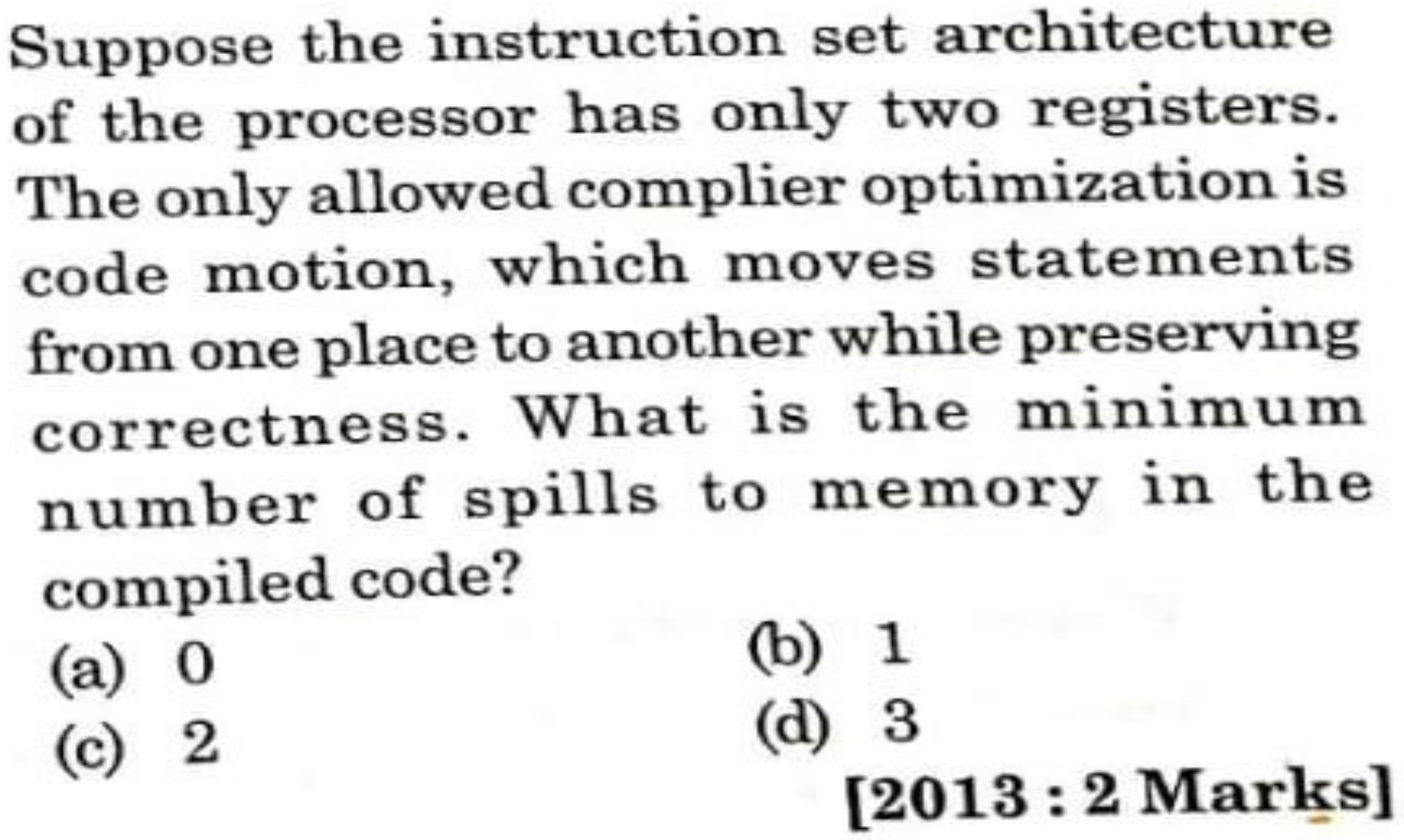


Common Data for Questions 4.4 and 4.5

The following code segment is executed on a processor which allows only register operands in its instructions. Each instruction can have almost two source operands and one destinations operand. Assume that all variables are dead after this code segment.

```
c = a + b;
d = c * a;
e = c + a;
x = c * c;
if (x > a) {
    y = a * a;
}
else {
    d = d * d;
    e = e * e;
}
```

Question(cont)



Suppose the instruction set architecture of the processor has only two registers. The only allowed complier optimization is code motion, which moves statements from one place to another while preserving correctness. What is the minimum number of spills to memory in the compiled code?

(a) 0

(b) 1

(c) 2

(d) 3

[2013 : 2 Marks]

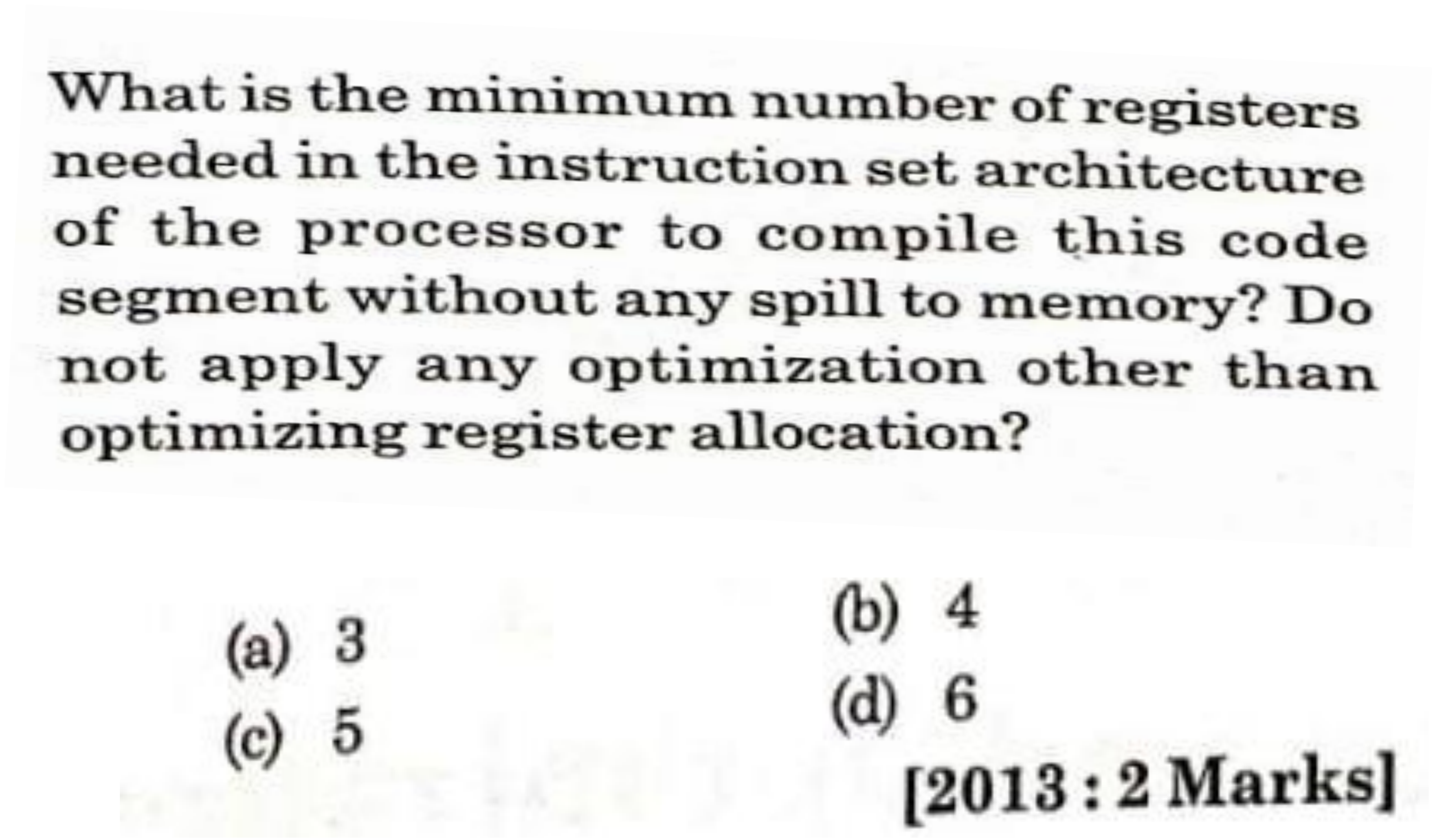
OPTION B

4 **(b)**

```
c = a + b;
x = c * c;                                    ...(i)
if (x > a) {
    y = a * a;
}
else {
    d = c * a;                               ...(ii)
    e = c + a;
    d = d * d;
    e = e * e;
}
```

(i) is to store c*c, it needs one memory spill.
(ii) is uses previous same(i) memory spill to store c*a.
Number of memory spills are used in above program is one. With the use of one memory cell and two registers the above optimized code can be executed.

Question



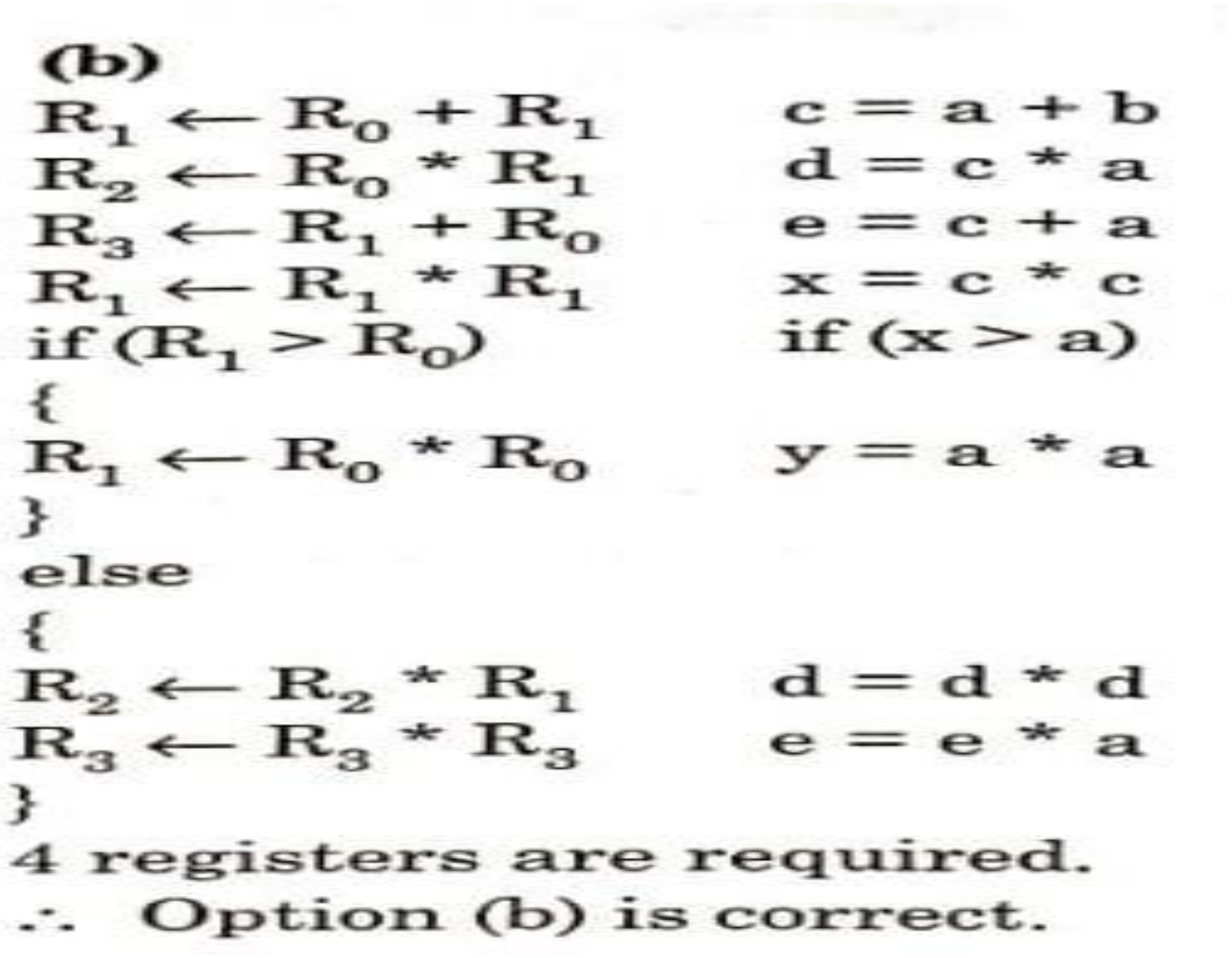
What is the minimum number of registers needed in the instruction set architecture of the processor to compile this code segment without any spill to memory? Do not apply any optimization other than optimizing register allocation?

(a) 3 (b) 4

(c) 5 (d) 6

[2013 : 2 Marks]

OPTION B



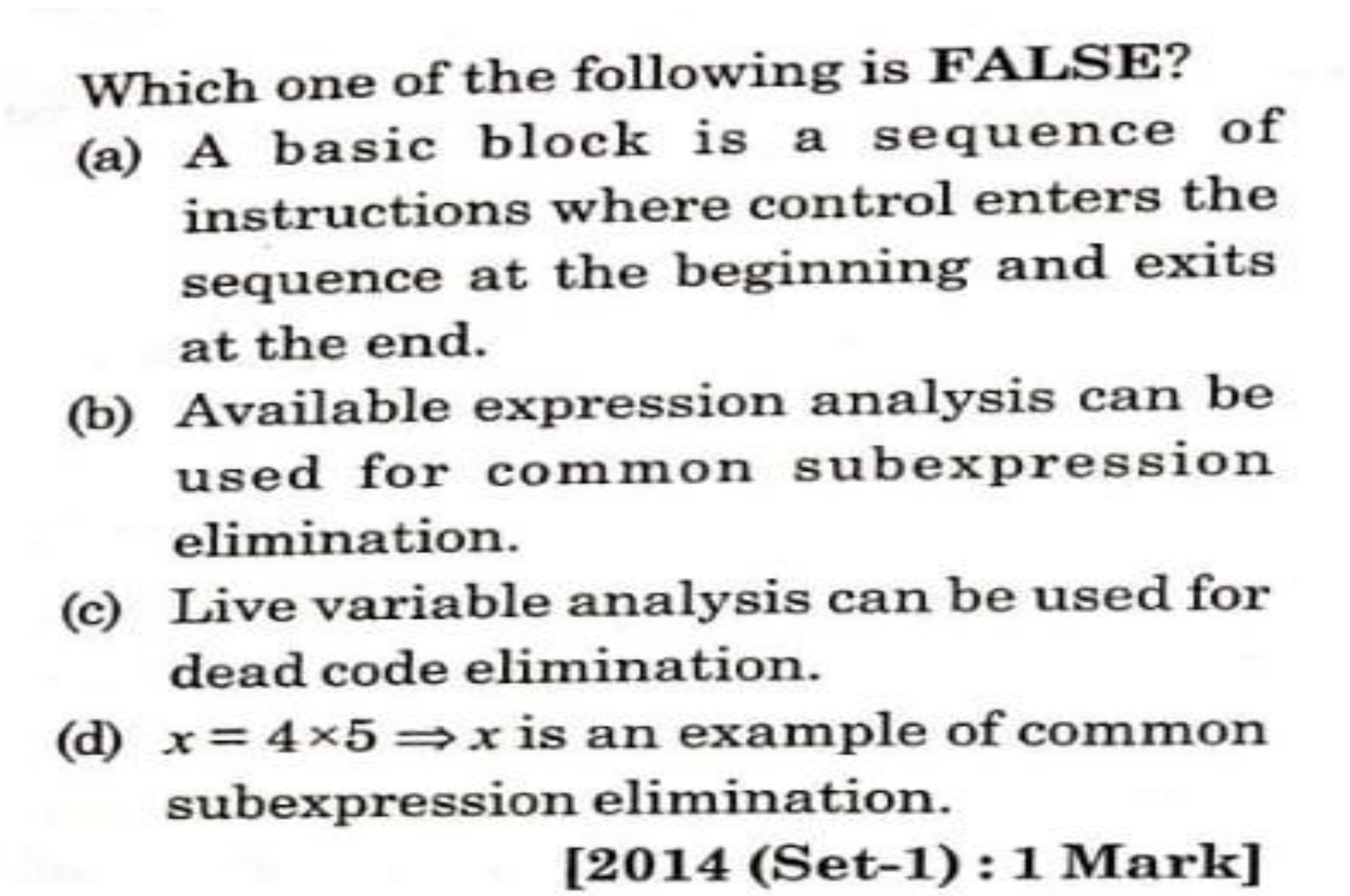
(b)

$R_1 \leftarrow R_0 + R_1$	$c = a + b$
$R_2 \leftarrow R_0 * R_1$	$d = c * a$
$R_3 \leftarrow R_1 + R_0$	$e = c + a$
$R_1 \leftarrow R_1 * R_1$	$x = c * c$
if $(R_1 > R_0)$	if $(x > a)$
{	
$R_1 \leftarrow R_0 * R_0$	$y = a * a$
}	
else	
{	
$R_2 \leftarrow R_2 * R_1$	$d = d * d$
$R_3 \leftarrow R_3 * R_3$	$e = e * a$
}	

4 registers are required.

$\therefore$ Option (b) is correct.

Question



Which one of the following is **FALSE**?

(a) A basic block is a sequence of instructions where control enters the sequence at the beginning and exits at the end.

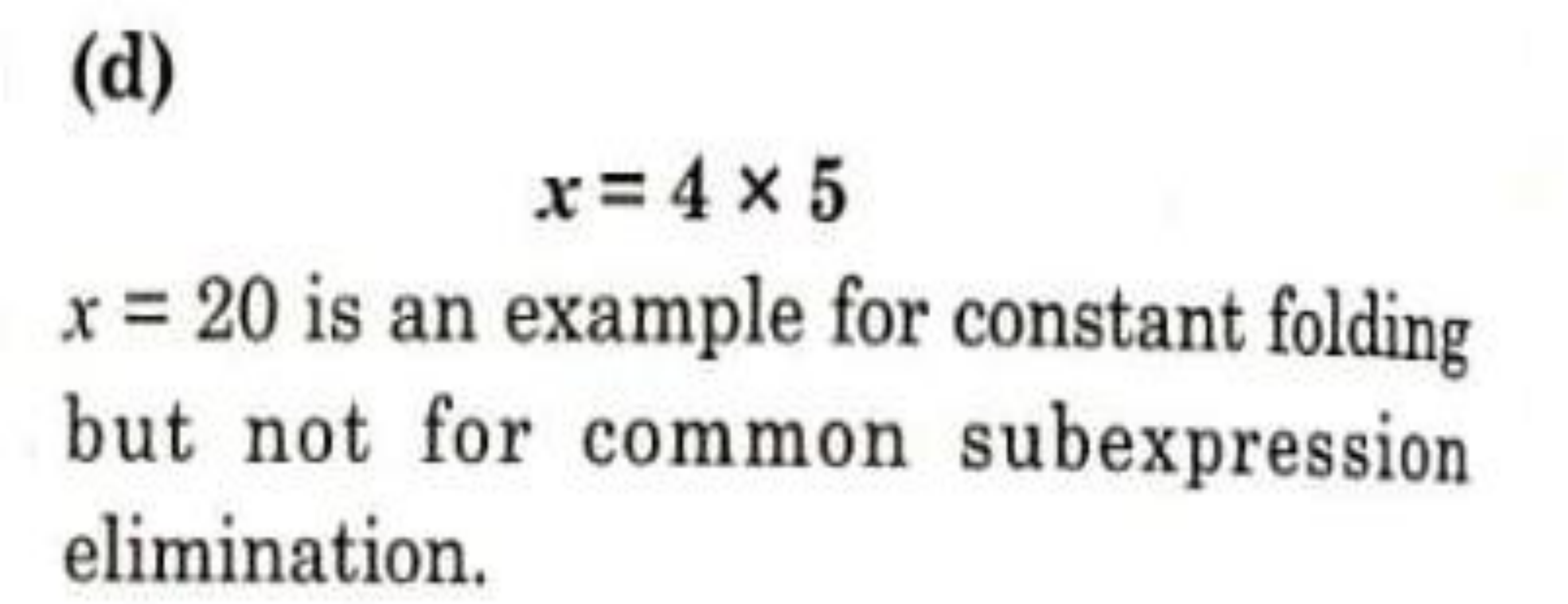
(b) Available expression analysis can be used for common subexpression elimination.

(c) Live variable analysis can be used for dead code elimination.

(d) $x = 4 \times 5 \Rightarrow x$ is an example of common subexpression elimination.

[2014 (Set-1) : 1 Mark]

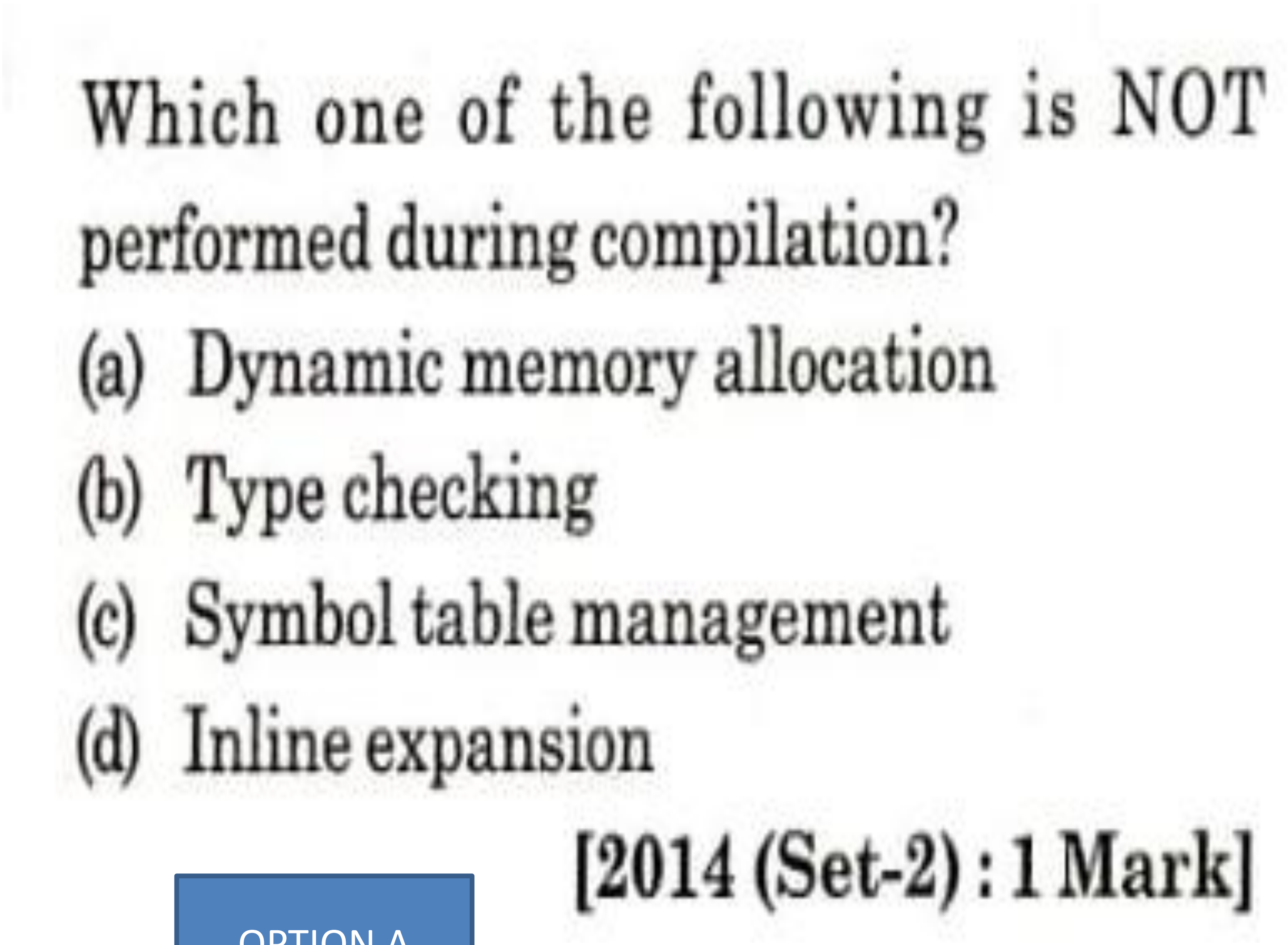
OPTION D


(d)

$$x = 4 \times 5$$

$x = 20$ is an example for constant folding but not for common subexpression elimination.

Question



Which one of the following is NOT performed during compilation?

(a) Dynamic memory allocation

(b) Type checking

(c) Symbol table management

(d) Inline expansion

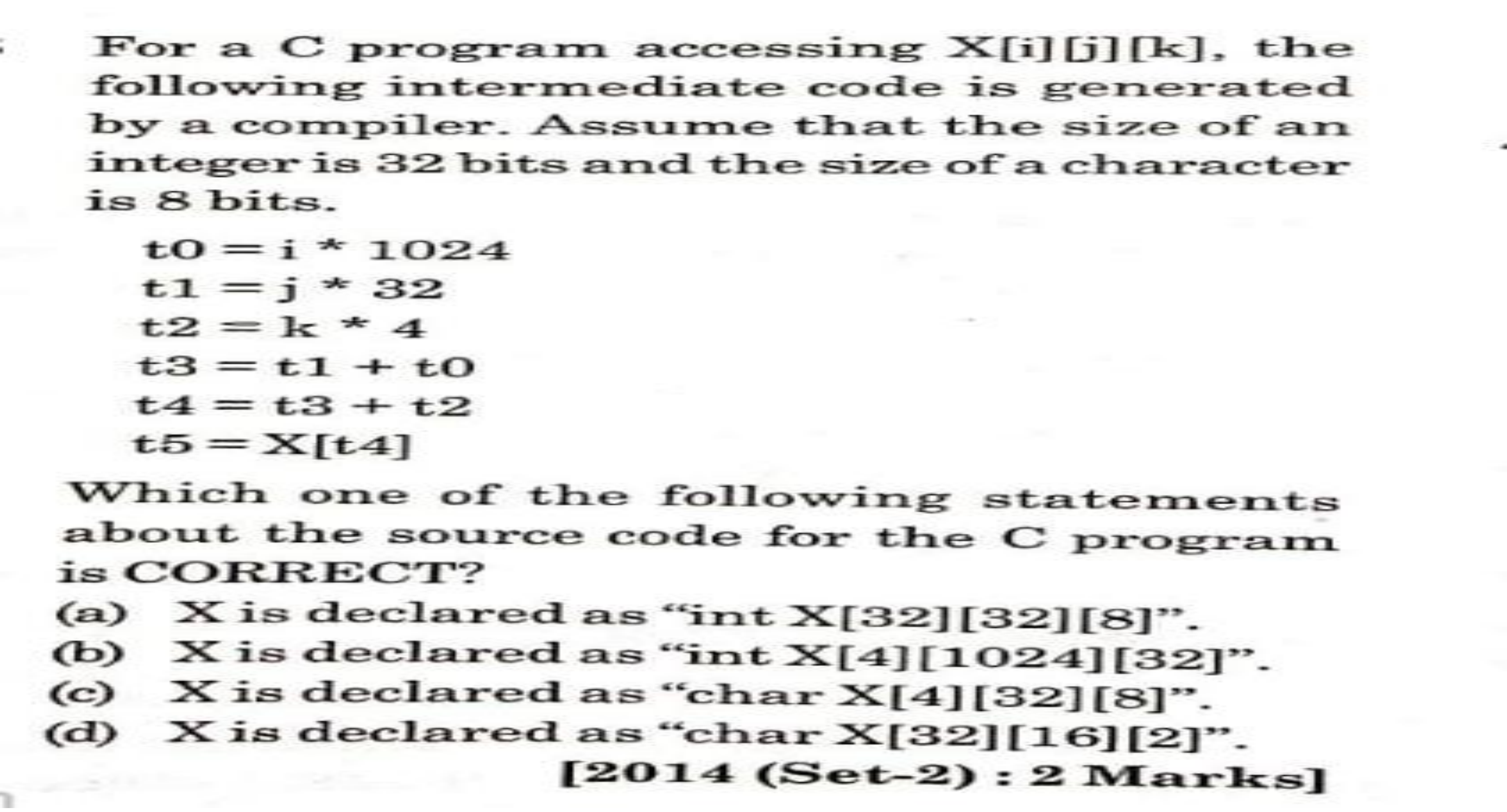
[2014 (Set-2) : 1 Mark]

OPTION A

(a)

Dynamic memory allocation performed during runtime whereas type checking, symbol table management and Inline expansion is performed during compilation.

Question



For a C program accessing X[i][j][k], the following intermediate code is generated by a compiler. Assume that the size of an integer is 32 bits and the size of a character is 8 bits.

```
t0 = i * 1024
t1 = j * 32
t2 = k * 4
t3 = t1 + t0
t4 = t3 + t2
t5 = X[t4]
```

Which one of the following statements about the source code for the C program is CORRECT?

(a) X is declared as "int X[32][32][8]".
(b) X is declared as "int X[4][1024][32]".
(c) X is declared as "char X[4][32][8]".
(d) X is declared as "char X[32][16][2]".

[2014 (Set-2) : 2 Marks]

OPTION A

(a)

Each integer element requires 4 bytes

int X [32] [32] [8];

To access X[i][j][k] :

X [(i*32*8+j*8 + k)*4] ...(1)

(integer size = 4)

From given intermediate code:

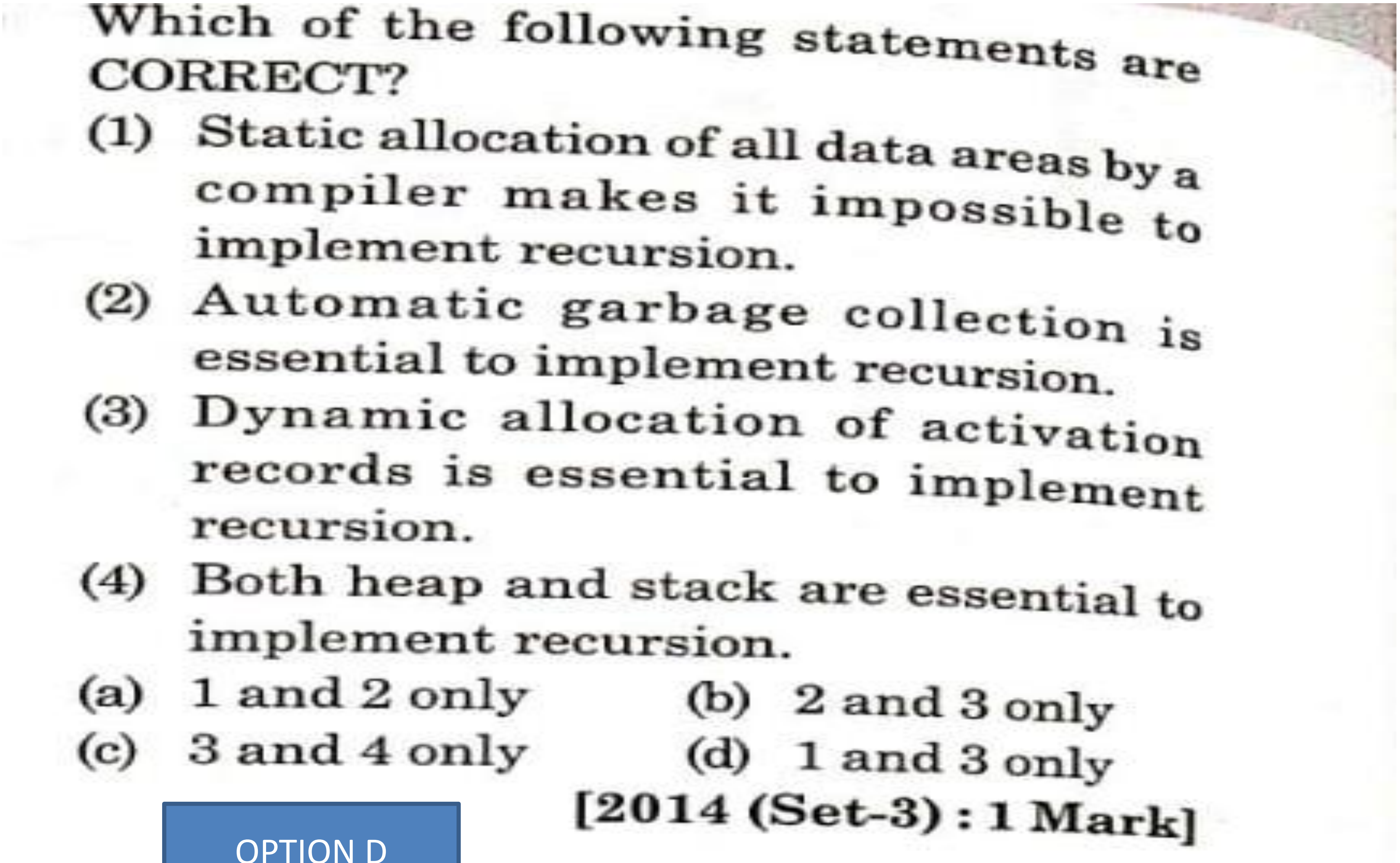
$X[t_4] = X[t_3 + t_2] = X[t_1 + t_0 + t_2]$

$= X[t_0 + t_1 + t_2]$

= X [(i*1024+j*32+k *4)] ...(2)

∴ (1) and (2) are equivalent.

Question



Which of the following statements are CORRECT?

(1) Static allocation of all data areas by a compiler makes it impossible to implement recursion.

(2) Automatic garbage collection is essential to implement recursion.

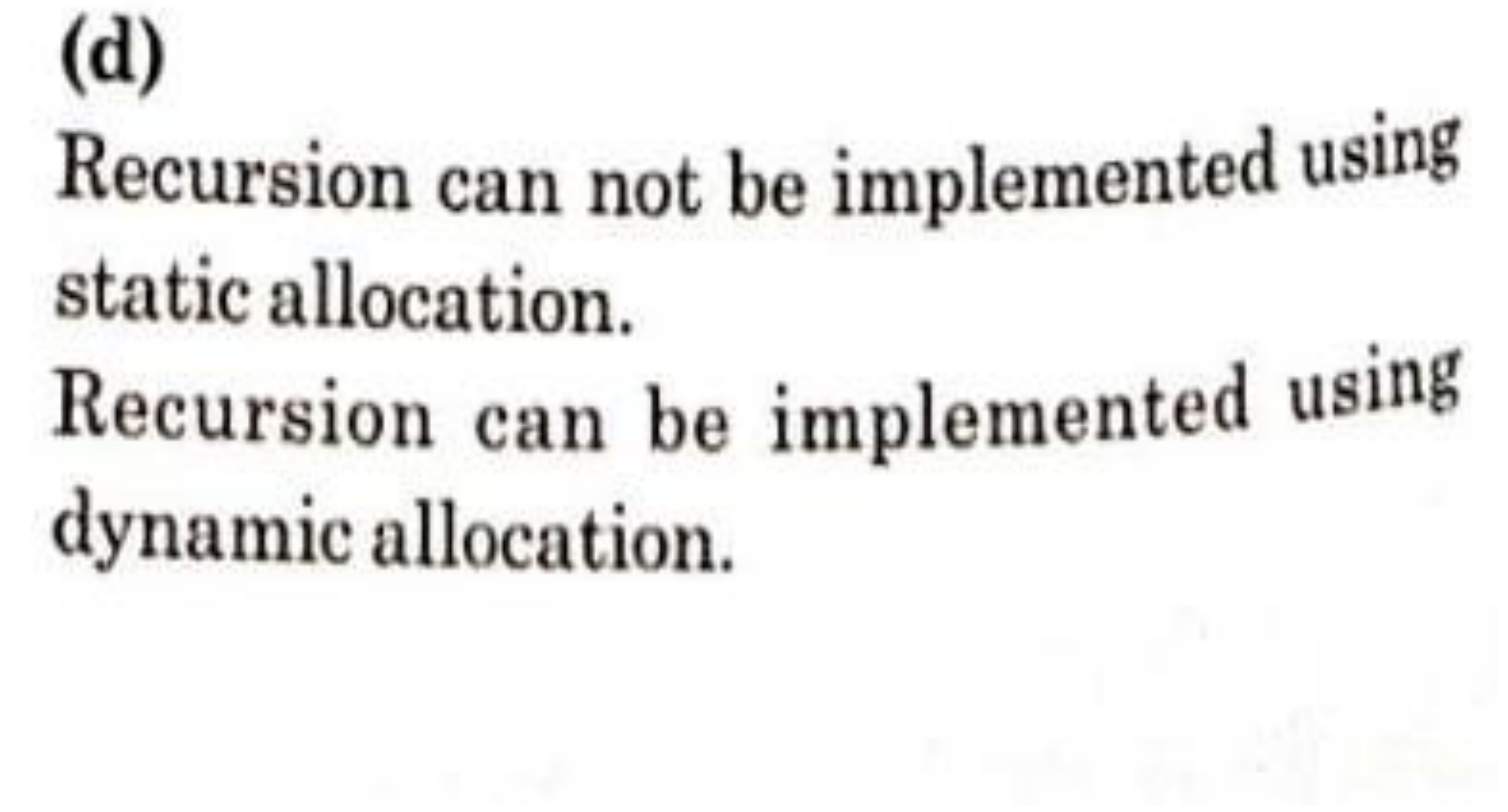
(3) Dynamic allocation of activation records is essential to implement recursion.

(4) Both heap and stack are essential to implement recursion.

(a)	1 and 2 only	(b)	2 and 3 only
(c)	3 and 4 only	(d)	1 and 3 only

[2014 (Set-3) : 1 Mark]

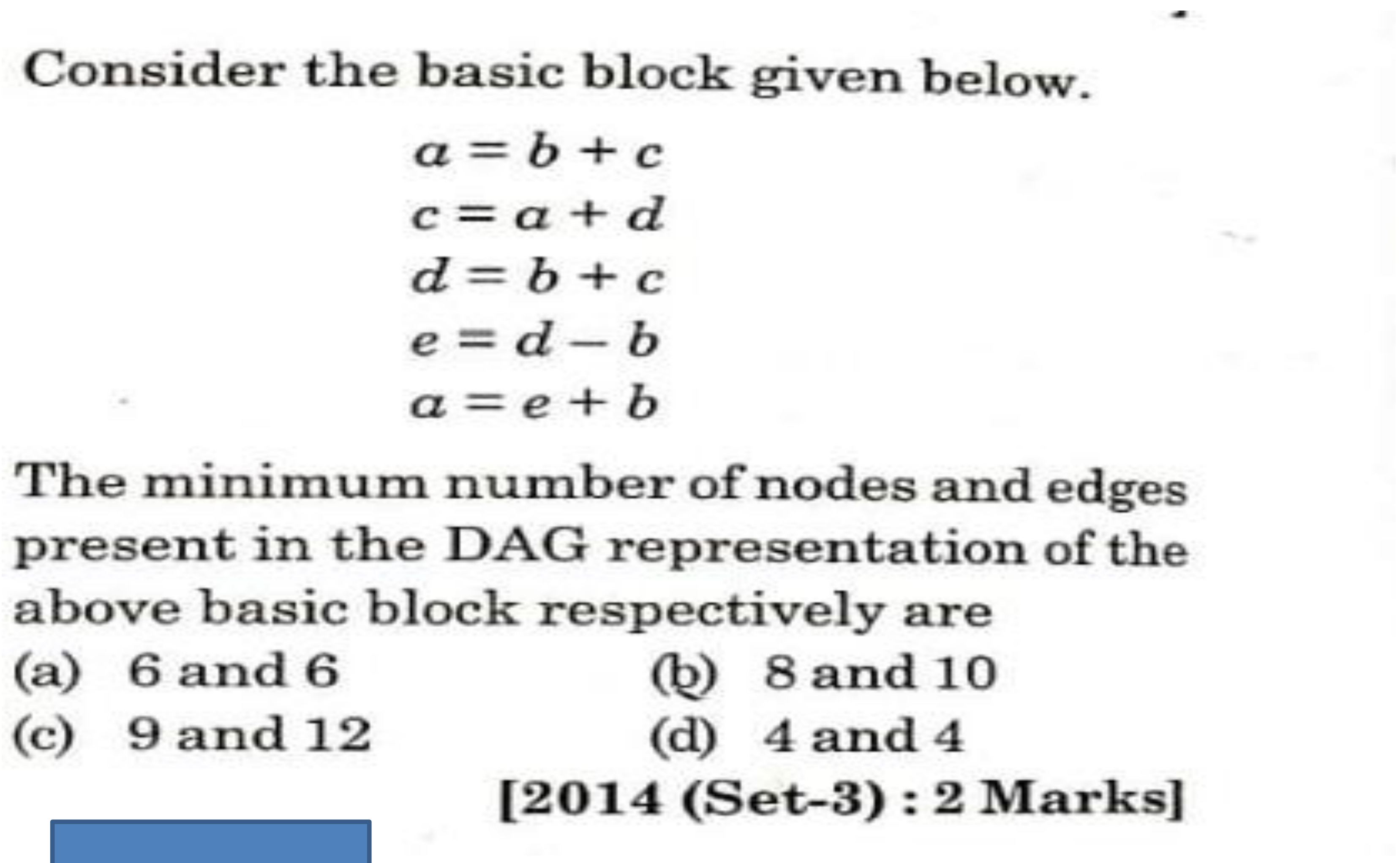
OPTION D


(d)

Recursion can not be implemented using static allocation.

Recursion can be implemented using dynamic allocation.

Question



Consider the basic block given below.

$$a = b + c$$
$$c = a + d$$
$$d = b + c$$
$$e = d - b$$
$$a = e + b$$

The minimum number of nodes and edges present in the DAG representation of the above basic block respectively are

(a) 6 and 6 (b) 8 and 10

(c) 9 and 12 (d) 4 and 4

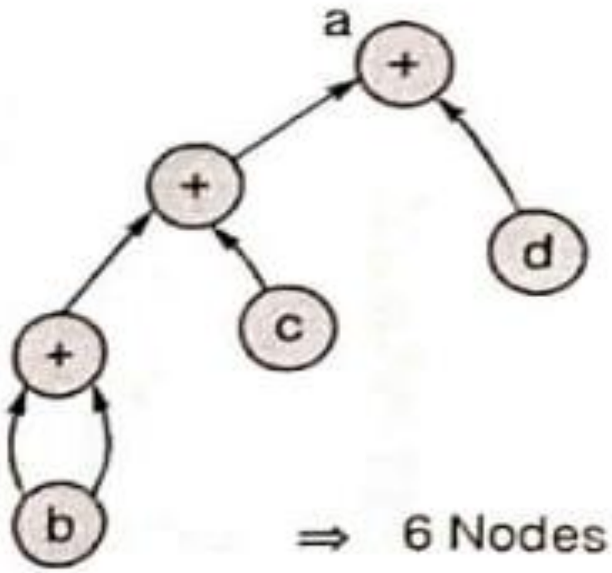
[2014 (Set-3) : 2 Marks]

OPTION A

(a)

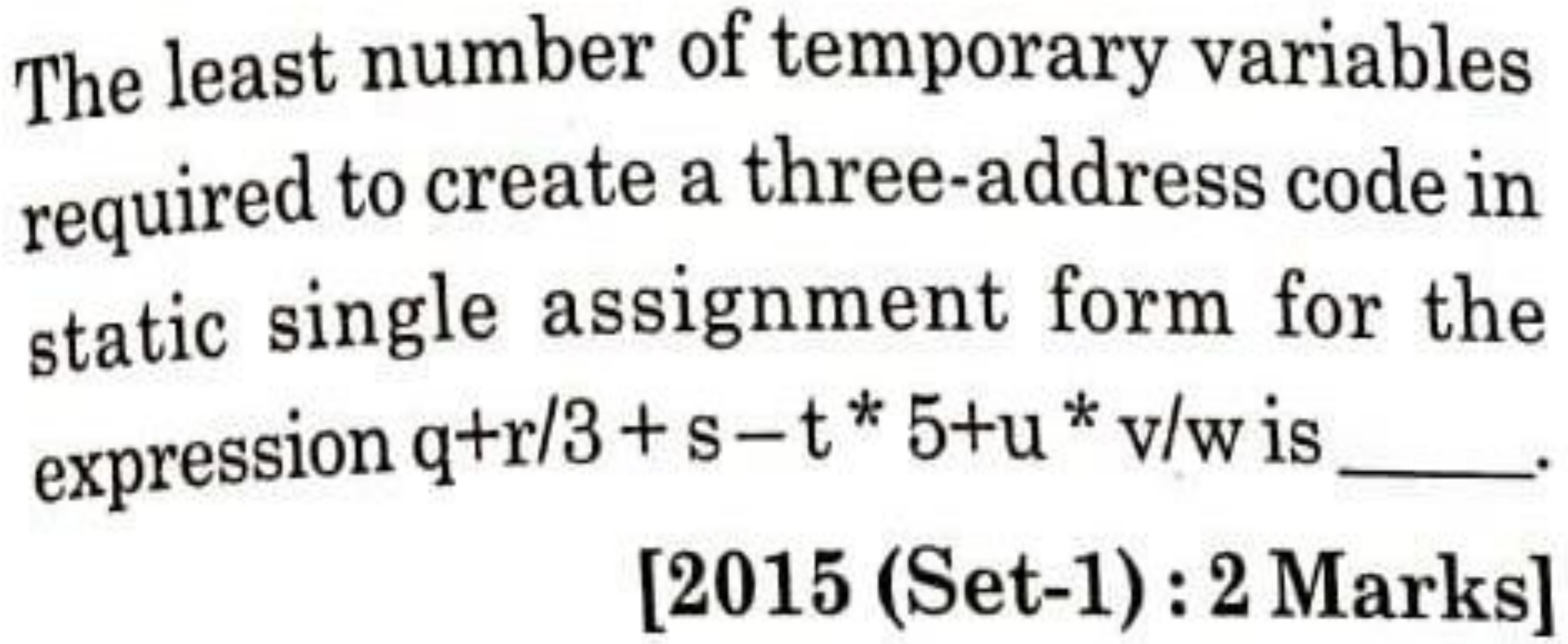
$$\left.\begin{array}{l} \underline{a} = b + c \\ \underline{c} = a + d \\ \underline{d} = b + c \\ \underline{e} = d - b \\ a = e + b \end{array}\right\} \begin{array}{l} \Rightarrow \quad a = e + b \\ \Rightarrow \quad a = d - b - b = d \\ \Rightarrow \quad a = b + c \\ \Rightarrow \quad a = b + a + d \\ \Rightarrow \quad a = b + b + c + d \end{array}$$

$\therefore\ b + b + c + d$ is expression



$\Rightarrow$ 6 Nodes and 6 edges

Question



The least number of temporary variables required to create a three-address code in static single assignment form for the expression q+r/3 + s – t * 5+u * v/w is ____.

[2015 (Set-1) : 2 Marks]

8

(8)

In static single assignment, every variable assigned only once and that variable can be used any number of times without assignment.

Expression: $q+r/3+s - t*5+u * v/w$

$t_1 = r / 3$

$t_2 = q + t_1$

$t_3 = t_2 + s$

$t_4 = t * 5$

$t_5 = t_3 - t_4$

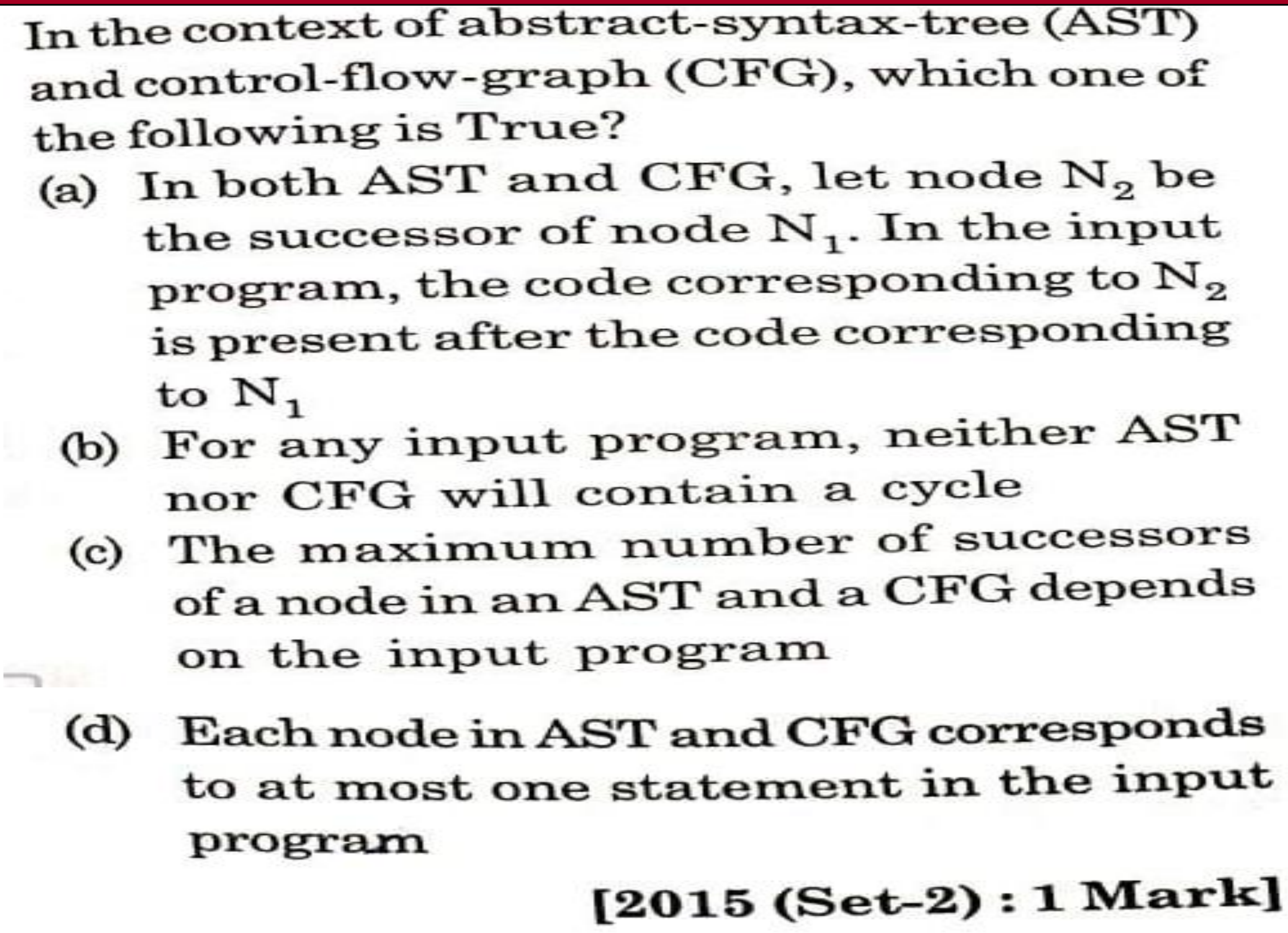
$t_6 = u * v$

$t_7 = t_6 / w$

$t_8 = t_5 + t_7$

$\therefore$ 8 temporary variables are required.

Question



In the context of abstract-syntax-tree (AST) and control-flow-graph (CFG), which one of the following is True?

(a) In both AST and CFG, let node N_2 be the successor of node N_1. In the input program, the code corresponding to N_2 is present after the code corresponding to N_1

(b) For any input program, neither AST nor CFG will contain a cycle

(c) The maximum number of successors of a node in an AST and a CFG depends on the input program

(d) Each node in AST and CFG corresponds to at most one statement in the input program

[2015 (Set-2) : 1 Mark]

OPTION C

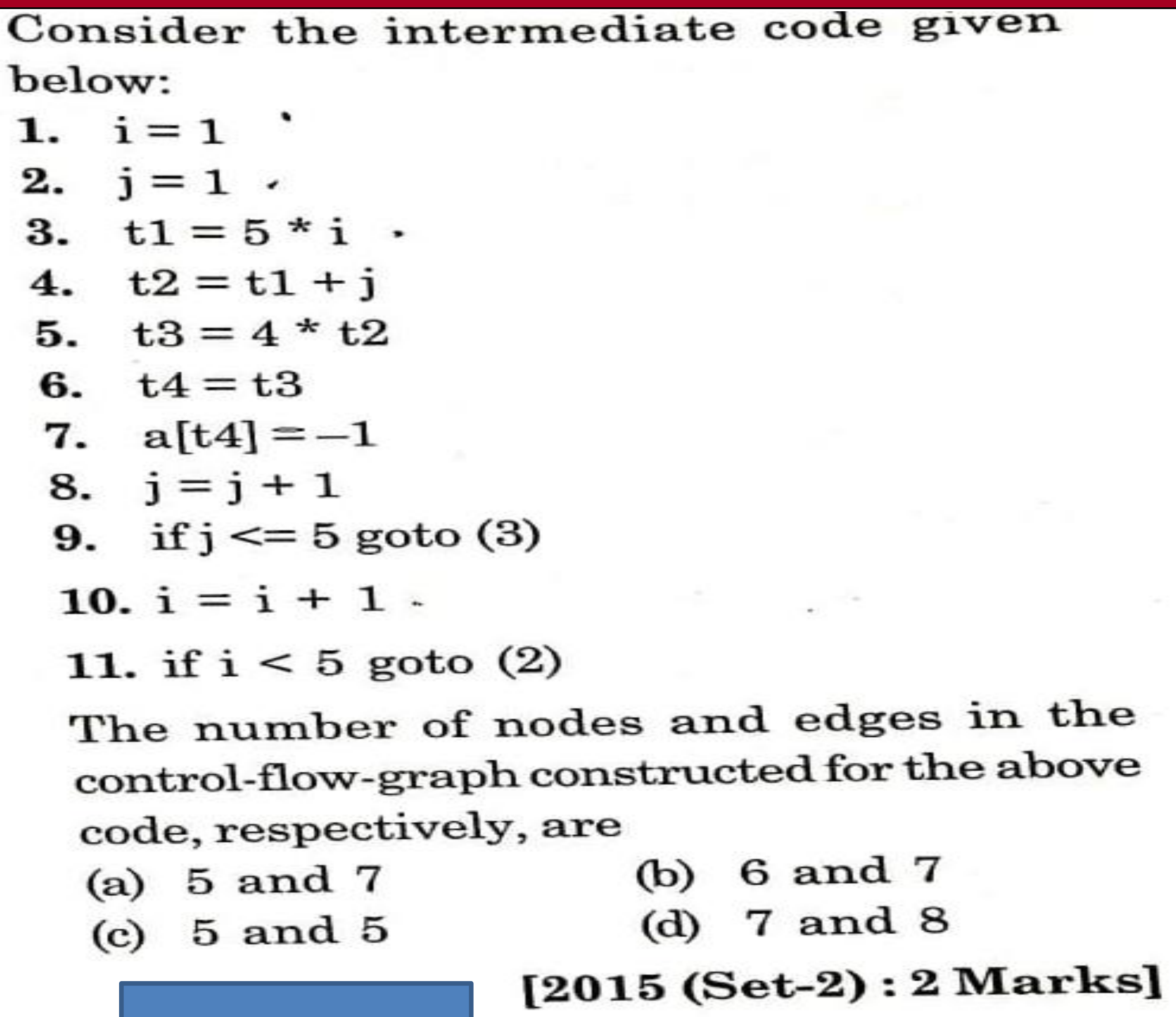
(c)

Maximum number of successors of a node in an AST and a CFG depends on input program.

CFG can contain a cycle.

A CFG node may corresponds to more than one statement.

Question



Consider the intermediate code given below:

```
1.  i = 1
2.  j = 1
3.  t1 = 5 * i
4.  t2 = t1 + j
5.  t3 = 4 * t2
6.  t4 = t3
7.  a[t4] = –1
8.  j = j + 1
9.  if j <= 5 goto (3)
10. i = i + 1
11. if i < 5 goto (2)
```

The number of nodes and edges in the control-flow-graph constructed for the above code, respectively, are

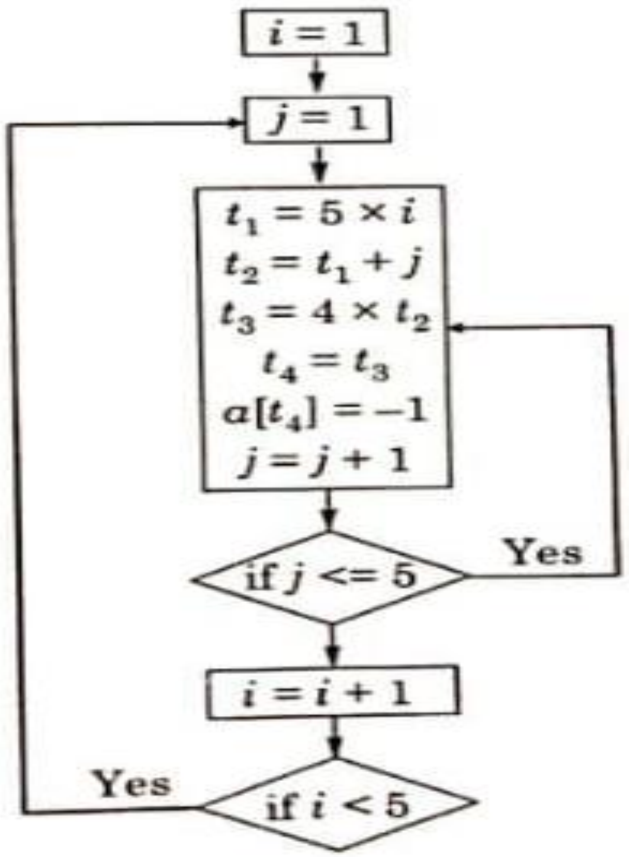
(a) 5 and 7 (b) 6 and 7

(c) 5 and 5 (d) 7 and 8

[2015 (Set-2) : 2 Marks]

OPTION B

(b)



Number of nodes = 6

Number of edges = 7

CODE OPTIMIZATION

Question

Which of the following comment about peephole optimization is true?

A)It is applied to a small part of the code and applied repeatedly

B)It can be used to optimize intermediate code

C)It can be applied to a portion of the code that is not contiguous

D)It is applied in the symbol table to optimize the memory requirements.

It is applied to a small part of the code and applied repeatedly

Question

Which of the following class of statement usually produces no executable code when compiled?

A)Declaration

B)Assignment statements

C)Input and output statements

D)Structural statements

A)Declaration

Question

Substitution of values for names (whose values are constants) is done in

A)Local optimization

B)Loop optimization

C)Constant folding

D)Strength reduction

C)Constant folding

Question

In compiler terminology reduction in strength means

A)Replacing run time computation by compile time computation

B)Removing loop invariant computation

C)Removing common subexpressions

D)Replacing a costly operation by a relatively cheaper one

Option D

Question

Which of the following statements about peephole optimization is False?

A)It is applied to a small part of the code

B)It can be used to optimize intermediate code

C)To get the best out of this, it has to be applied repeatedly

D)It can be applied to the portion of the code that is not contiguous

It can be applied to the portion of the code that is not contiguous

Question

The graph that shows basic blocks and their successor relationship is called:

A)DAG

B)Control graph

C)Flow graph

D)Hamiltonian graph

Flow graph

Question

In compiler optimization, operator strength reduction uses mathematical identities to replace slow math operations with faster operations. Which of the following code replacements is an illustration of operator strength reduction ?

- A)Replace P + P by 2 * P or Replace 3 + 4 by 7.
- B)Replace P * 32 by P << 5
- C)Replace P * 0 by 0
- D)Replace (P << 4) – P by P * 15
- Replace P * 32 by P << 5

Question

In ________, the bodies of the two loops are merged together to form a single loop provided that they do not make any references to each other.

- A)Loop unrolling
- B)Strength reduction
- C)Loop concatenation
- D)Loop jamming
- Loop jamming

Question

Loop unrolling is a code optimization technique:

A)That avoids tests at every iteration of the loop.

B)That improves performance by decreasing the number of instructions in a basic block.

C)That exchanges inner loops with outer loops

D)That reorders operations to allow multiple computations to happen in parallel

- That avoids tests at every iteration of the loop

Question

Peer-hole optimization is a form of :

A)Loop optimization

B)Local optimization

C)Constant folding

D)Data flow analysis

B)Local optimization

Question

Dead-code elimination in machine code optimization refers to :

- A) Removal of all labels.
- B) Removal of values that never get used.
- C) Removal of function which are not involved.
- D) Removal of a module after its use.
- Removal of values that never get used.

Question

In the context of compiler design, “reduction in strength” refers to :

- A)Code optimization obtained by the use of cheaper machine instructions
- B)Reduction in accuracy of the output
- C)Reduction in the range of values of input variables
- D)Reduction in efficiency of the program
- Code optimization obtained by the use of cheaper machine instructions

Question

Some code optimizations are carried out on the intermediate code because:

A)They enhance the portability of the compiler to other target processors

B)Program analysis is more accurate on intermediate code than on machine code

C)The information from dataflow analysis cannot otherwise be used for optimization

D)The information from the front end cannot otherwise be used for optimization

A and B are true

Question

Which one of the following is FALSE?

A)A basic block is a sequence of instructions where control enters the sequence at the beginning and exits at the end.

B)Available expression analysis can be used for common sub expression elimination.

C)Live variable analysis can be used for dead code elimination.

D)x = 4 ∗ 5 => x = 20 is an example of common subexpression elimination.

Answer is D

Question

One of the purposes of using intermediate code in compilers is to

A)make parsing and semantic analysis simpler.

B)improve error recovery and error reporting.

C)increase the chances of reusing the machine-independent code

optimizer in other compilers.

D)improve the register allocation.

- Option is C

Question

Consider the following C code segment.

for (i = 0, i<n; i++){

- for (j=0; j<n; j++) {
- if (i%2) {
- x += (4*j + 5*i);
- y += (7 + 4*j); } }}

Which one of the following is false?

A)The code contains loop invariant computation

B)There is scope of common sub-expression elimination in this code

C)There is scope of strength reduction in this code

D)There is scope of dead code elimination in this code

Option is D

QUESTION

Code optimisation is responsibility of:

A.Application programmer

B.Syatem programmer

C.Operating System

D.All of the above

Option D

SYMBOL TABLE

QUESTION

Select a Machine Independent phase of the compiler

a) Syntax Analysis

b) Intermediate Code generation

c) Lexical Analysis

d) All of the mentioned

Answer: d

Explanation: All of them work independent of a machine

QUESTION

By whom is the symbol table created?

a) Compiler

b) Interpreter

c) Assembler

d) None of the mentioned

Answer: a

Explanation: Symbol table is created by the compiler which contains the list of lexemes or tokens.

QUESTION

Which of these is not true about Symbol Table?

a) All the labels of the instructions are symbols

b) Table has entry for symbol name address value

c) Perform the processing of the assembler directives

d) Created during pass 1

Answer: c

Explanation: The Symbol table does not ever perform the processing of the assembler derivative

QUESTION

Which of these features of assembler are Machine-Dependent

a) Instruction formats

b) Addressing modes

c) Program relocation

d) All of the mentioned

Answer: d

Explanation: All of these options are features of assembler which are machine dependent

QUESTION

The symbol table implementation is based on the property of locality of reference is ___________

A. Linear list

B. Search tree

C. Hash table

D. Self Organization list

Correct Answer : OPTION C, Hash table

QUESTION

Access time of the Symbol table will be logarithmic, if its implemented by ___________

A. Linear list

B. Search tree

C. Hash table

D. Self Organization list

Correct Answer : OPTION B, Search tree

QUESTION

Which table is permanent databases that has an entry for each terminal symbol?

A. Terminal table

B. Literal table

C. Identifier table

D. Reductions

Correct Answer : OPTION A, Terminal table

QUESTION

Symbol table can be used for:

A. Checking type compatibility

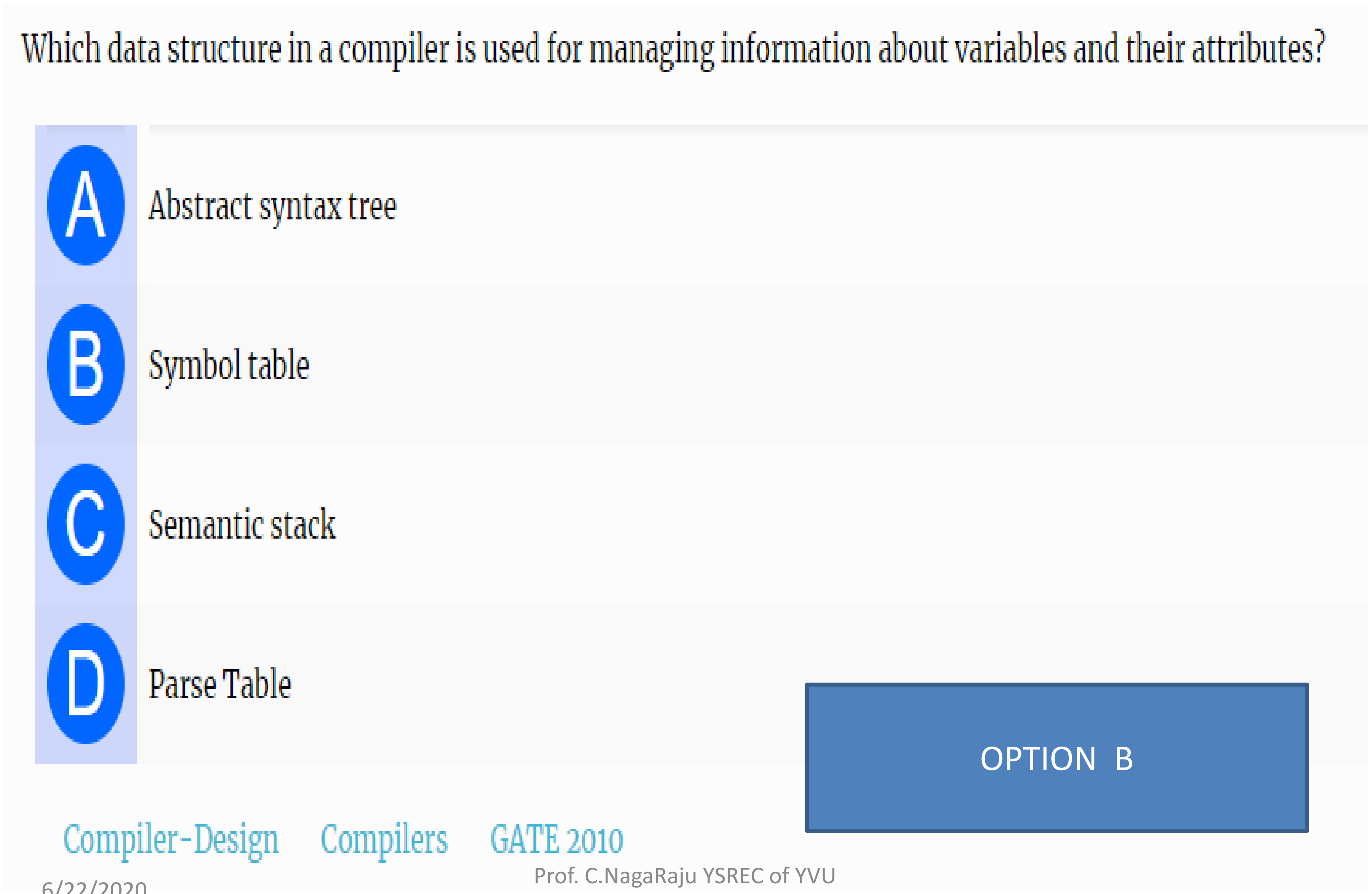
B. Suppressing duplication of error messages

C. Storage allocation

D. All of these

Option D

QUESTION



Which data structure in a compiler is used for managing information about variables and their attributes?

A Abstract syntax tree

B Symbol table

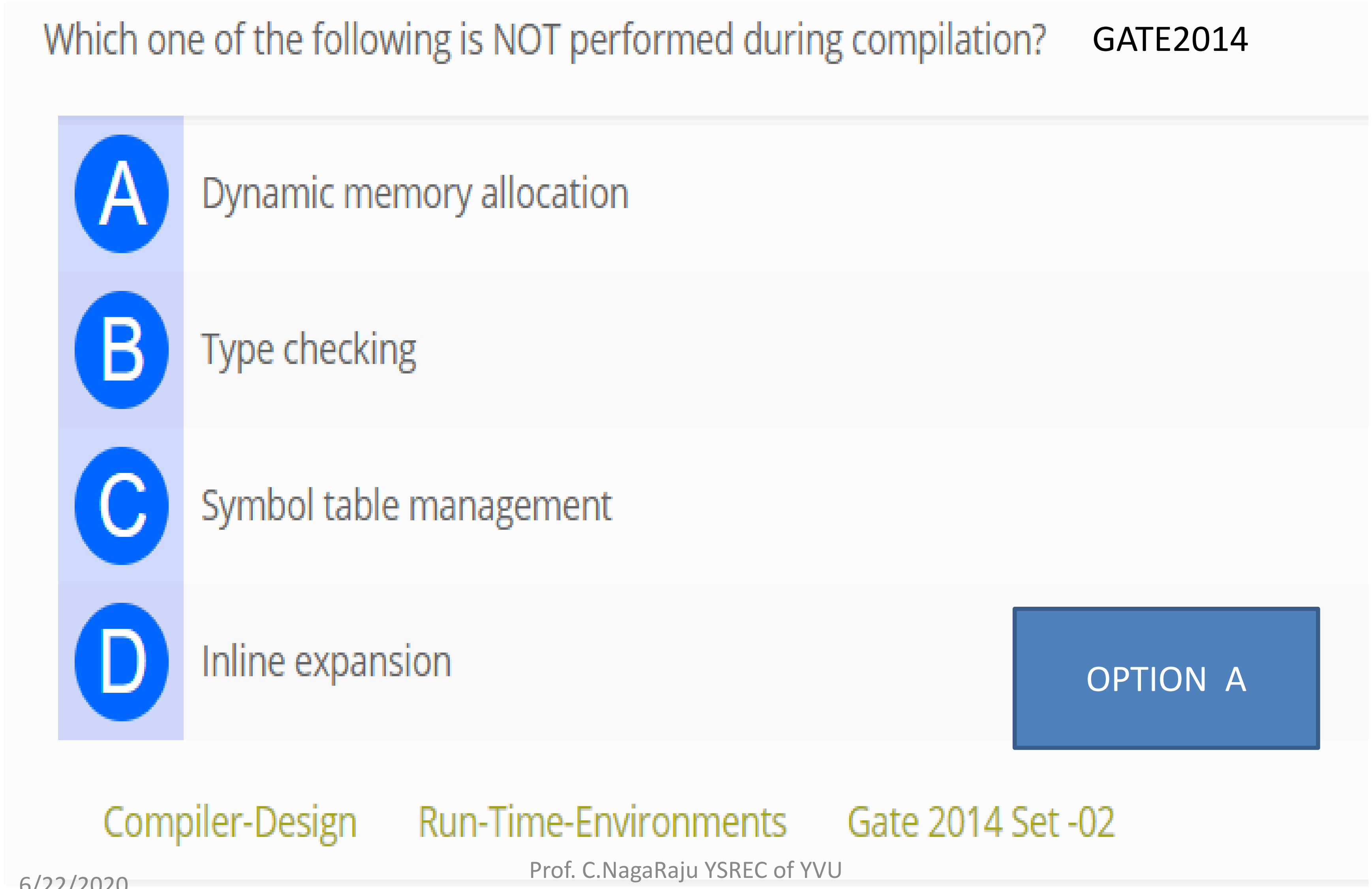
C Semantic stack

D Parse Table

Compiler-Design Compilers GATE 2010

OPTION B

QUESTION



Which one of the following is NOT performed during compilation? GATE2014

A Dynamic memory allocation

B Type checking

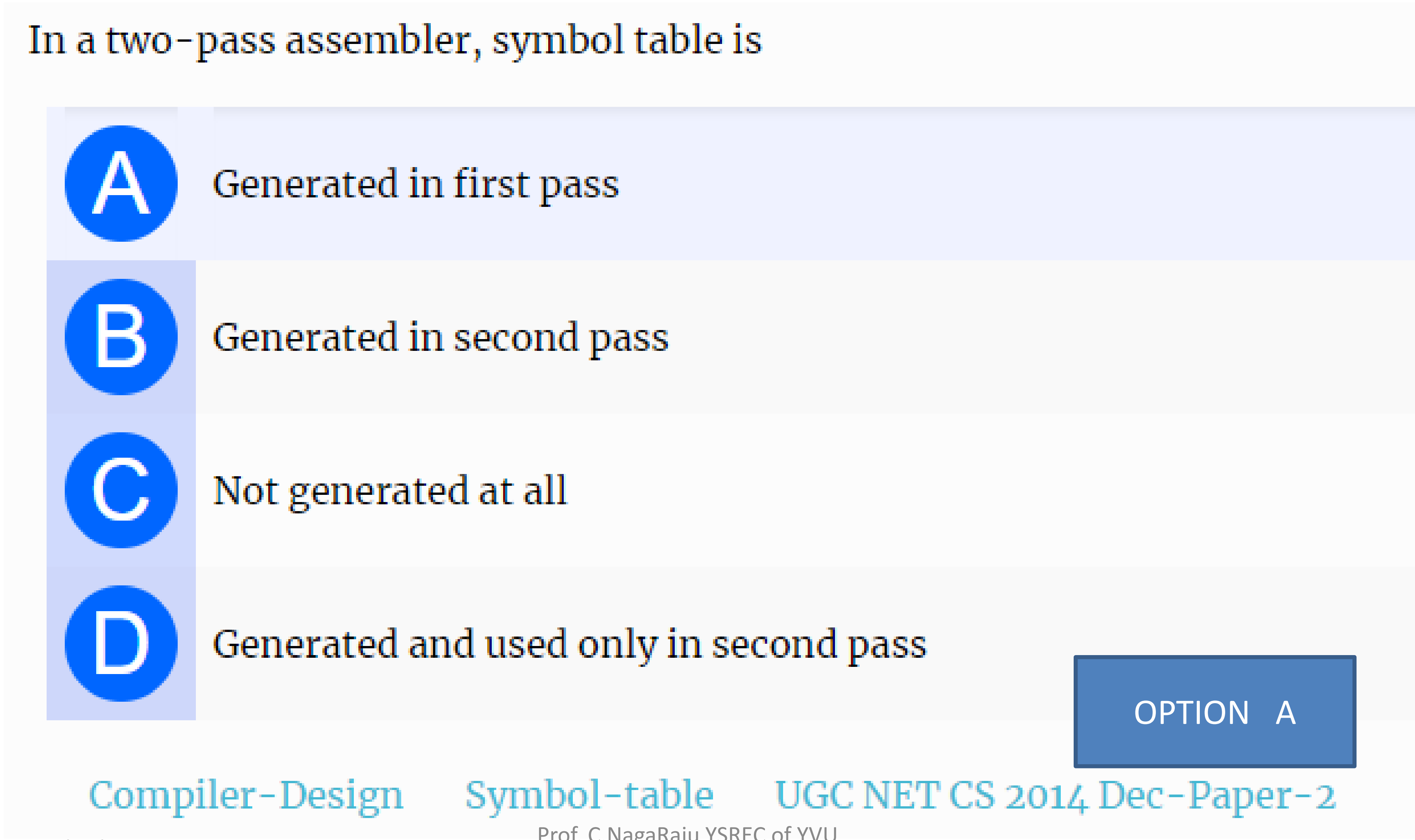
C Symbol table management

D Inline expansion

OPTION A

Compiler-Design Run-Time-Environments Gate 2014 Set -02

QUESTION



In a two-pass assembler, symbol table is

- A Generated in first pass
- B Generated in second pass
- C Not generated at all
- D Generated and used only in second pass

Compiler-Design Symbol-table UGC NET CS 2014 Dec-Paper-2

OPTION A

QUESTION

Uniform symbol table

A Contains all constants in the program

B Is a permanent table of division rules in the form of patterns for matching with the uniform symbol table to discover syntactic structure

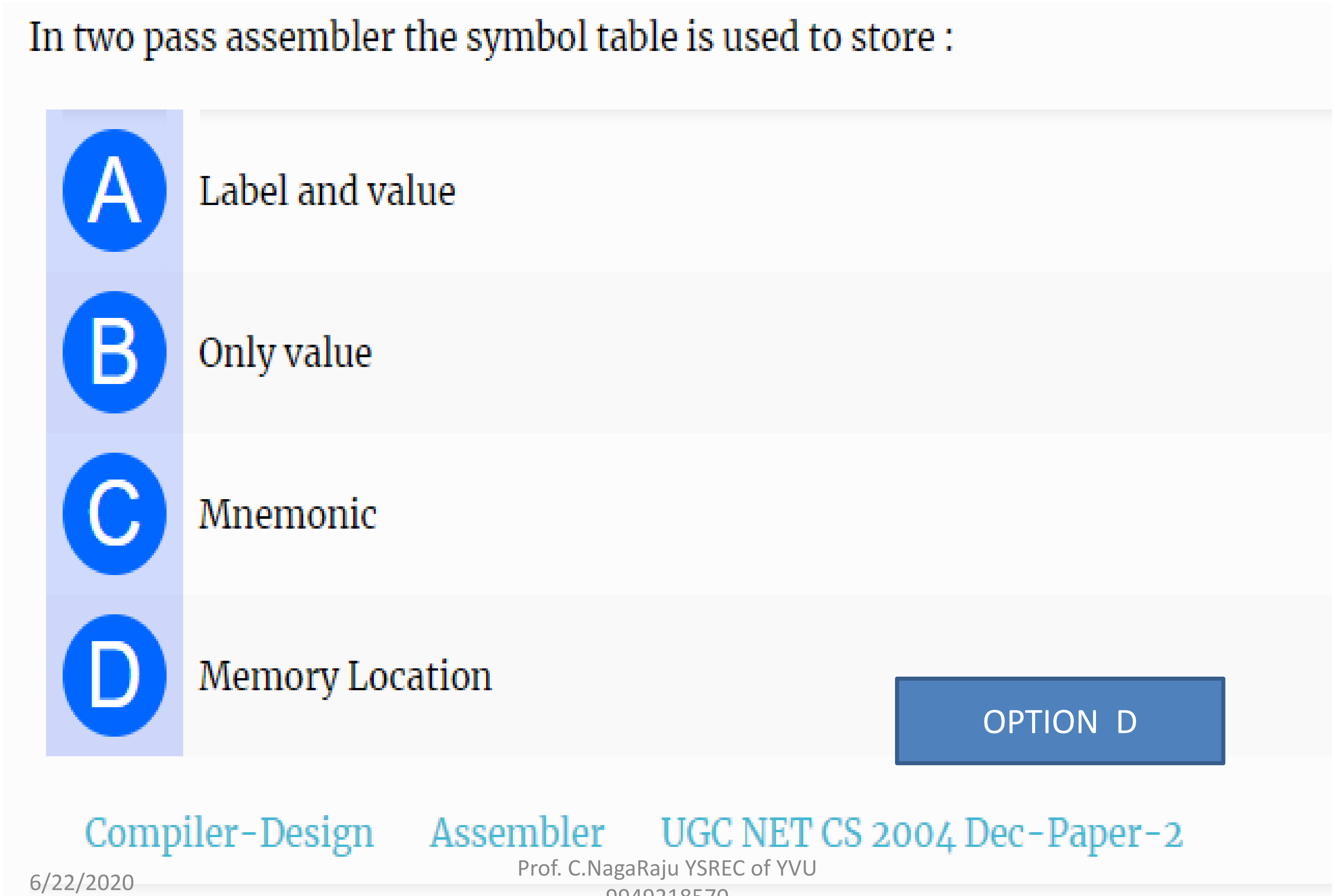
C Consists of full or partial list of the tokens as they appear in the program, created by Lexical Analysis and used for syntax analysis and interpretation

D A permanent table which lists all key words and special symbols of the language in symbolic form

Compiler-Design Symbol-Table TNPSC-2012-Polytechnic-CS

OPTION C

QUESTION



In two pass assembler the symbol table is used to store :

A Label and value

B Only value

C Mnemonic

D Memory Location

OPTION D

Compiler-Design Assembler UGC NET CS 2004 Dec-Paper-2

RUN TIME ENVIRONMENT

QUESTION

The identification of common sub-expression and replacement of run-time computations by compile-time computations is

A.local optimization

B.loop optimization

C.constant folding

D.data flow analysis

Option: C

QUESTION

The graph that shows basic blocks and their successor relationship is called

A.DAG

B.Flow graph

C.control graph

D.Hamiltonion graph

Option: B

QUESTION

The specific task storage manager performs

A.allocation/ deallocation of storage to programs

B.protection of storage area allocated to a program from illegal access by other programs in the system

C.the status of each program

D.both (a) and (b)

Option: D

QUESTION

When a computer is first turned on or resrarted, a special type of absolute loader is executed called

A." Compile and GO " loader

B.Boot loader

C.Boot strap loader

D.Relating loader

Option: C

QUESTION

Relocation bits used by relocating loader are specified by

A.relocating loader itself

B.linker

C.assembler

D.macro processor

Option: B

QUESTION

In some programming languages, an identifier is permitted to be a letter followed by any number of letters or digits. If L and D denotes the sets of letters and digits respectively, which of the following expressions define an identifier ?

A.(L∪ D) *

B.L (L ∪ D)*

C.(L . D)*

D.L . (L . D)*

Option: B

QUESTION

A language L allows declaration of arrays whose sizes are not known during compilation. It is required to make efficient use of memory. Which one of the following is true ?

A.a compiler using static memory allocation can be written for L

B.a compiler cannot be written for L ; an interpreter must be used

C.a compiler using dynamic memory allocation can be written for L

D.none of these

Option: C

THANK YOU