CMR
INSTITUTE OF
TECHNOLOGY

USN | | | | | | | | | | |

Improvement Test, May 2017

Sub:	Compiler Design							Code:	10CS63
Date:	30 / 05 / 2017	Duration:	90 mins	Max Marks:	50	Sem:	**VI**	Branch:	**CSE**

	Answer any 5 full questions	**Marks**	**OBE**	
			CO	**RBT**
1.a)	With a neat diagram, describe the general structure of an activation record.	[6]	CO6	L1
b)	What is heap? What are the desire properties of memory manager?	[4]	CO6	L2
2.	Explain briefly the performance metrics to be considered while designing a garbage collector.	[10]	CO6	L2
3.	Explain the code generation algorithm and generate code for the following expression. X= (a-b) * (a-c) Y= *p + X	[10]	CO6	L3
4.	What do you mean by calling sequence? Explain the actions performed during (a) Function Call (b) Return	[10]	CO6	L2
5.a)	Explain the strategy for reducing fragmentation in heap memory.	[5]	CO6	L2

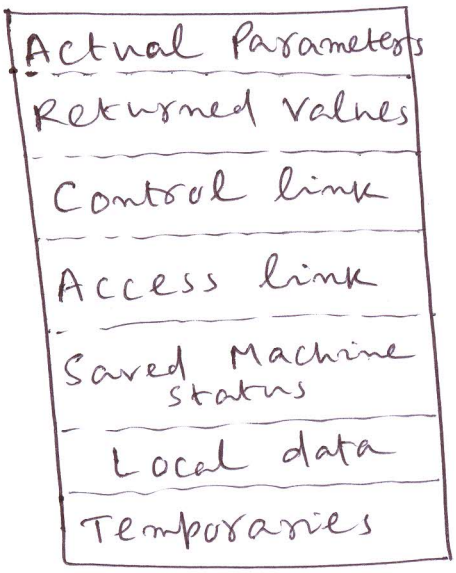
b.	Explain the following code optimization with example: i) Finding local common sub expression ii) Dead Code elimination	[5]	CO6	L2
6.	A C-code to compute Fibonacci numbers recursively is shown below: int f(int n) { int T,S; if(n<=2) return 1; S=f(n-1); T=f(n-2); return(S+T); } (a) Draw the activation tree for the call : f(7) (b) What is the largest no.of activation records that ever appear together on the stack?	[9+1]	CO6	L3

IMPROVEMENT TEST, May 2017

Scheme & Solution

Sub: Compiler Design, code: 10CS63

1.(a) General structure of an activation Record



(1M)

→ Explanation of each field (5M)

1.(b) Heap (1M)

→ Heap is the portion of the store that is used for data that lives indefinitely or until the program explicitly deletes it.

Desire Properties of Memory Manager (3M)

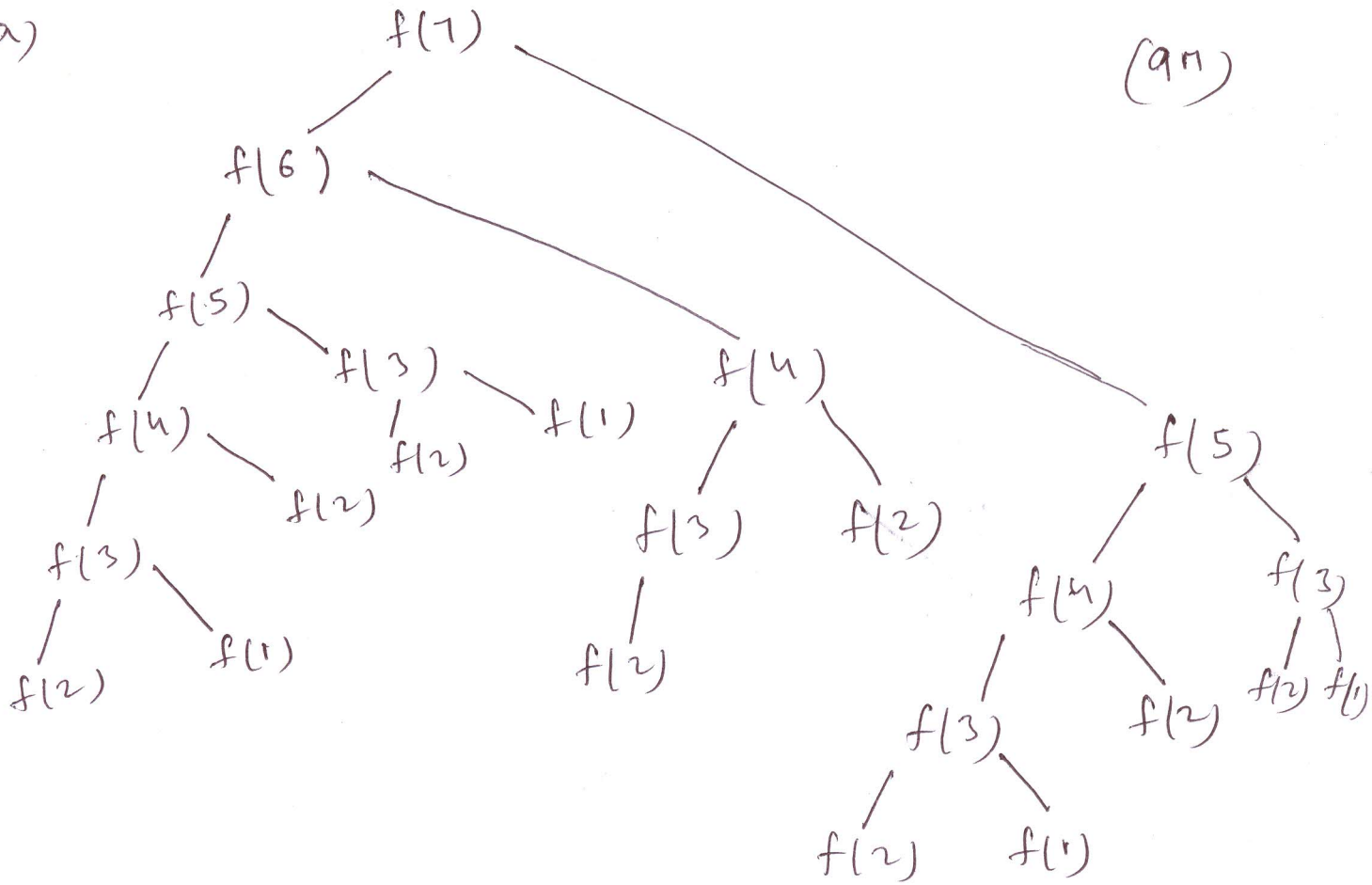
① Space efficiency

② Program efficiency

③ Low overhead

→ Explanation of above 3 points.

2. (a)



(9M)

(b) Largest no. of activation records that appear together on the stack is 6. (1M)

3. Performance metrics for designing a garbage collector

① Overall Execution time
② Space Usage
③ Pause Time
④ Program locality

(2.5 x 4 = 10M)

→ Explanation of above 4 points.

4. Code generation algorithm (5M) (2)

→ The function getReg(I) which selects registers for each memory location associated with the 3-add inst^n I.

Ex $x = y + z$

1. Use getReg($x = y + z$) to select registers for x, y and z. Call these R_x, R_y & R_z.
2. If y is not in R_y, then issue an inst^n LD R_y, y'
3. If z is not in R_z, then issue an inst^n LD R_z, z'
4. Issue the inst^n ADD R_x, R_y, R_z.

Code for the expression (5M)

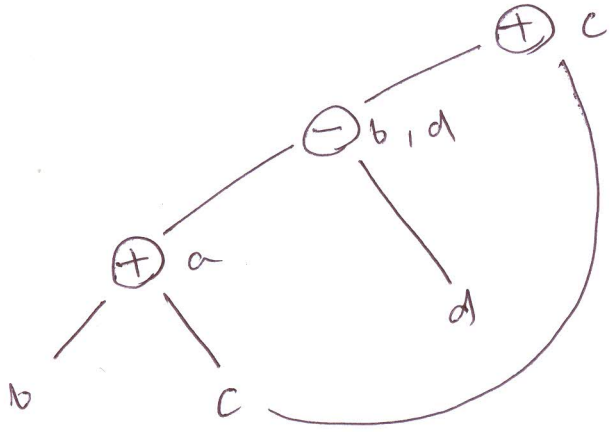
$$X = (a - b) * (a - c)$$
$$Y = *p + X$$

```
LD  R1, a
LD  R2, b
SUB R2, R1, R2
LD  R3, c
SUB R3, R1, R3
MUL R2, R2, R3
ST  X, R2
```

```
LD  R1, p
LD  R3, 0(R1)
ADD R3, R3, R2
ST  Y, R3
```

5. (a) Finding local common sub expression (2-5M)

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



Optimized code

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

(ii) Dead code elimination (2-5M)

Ex

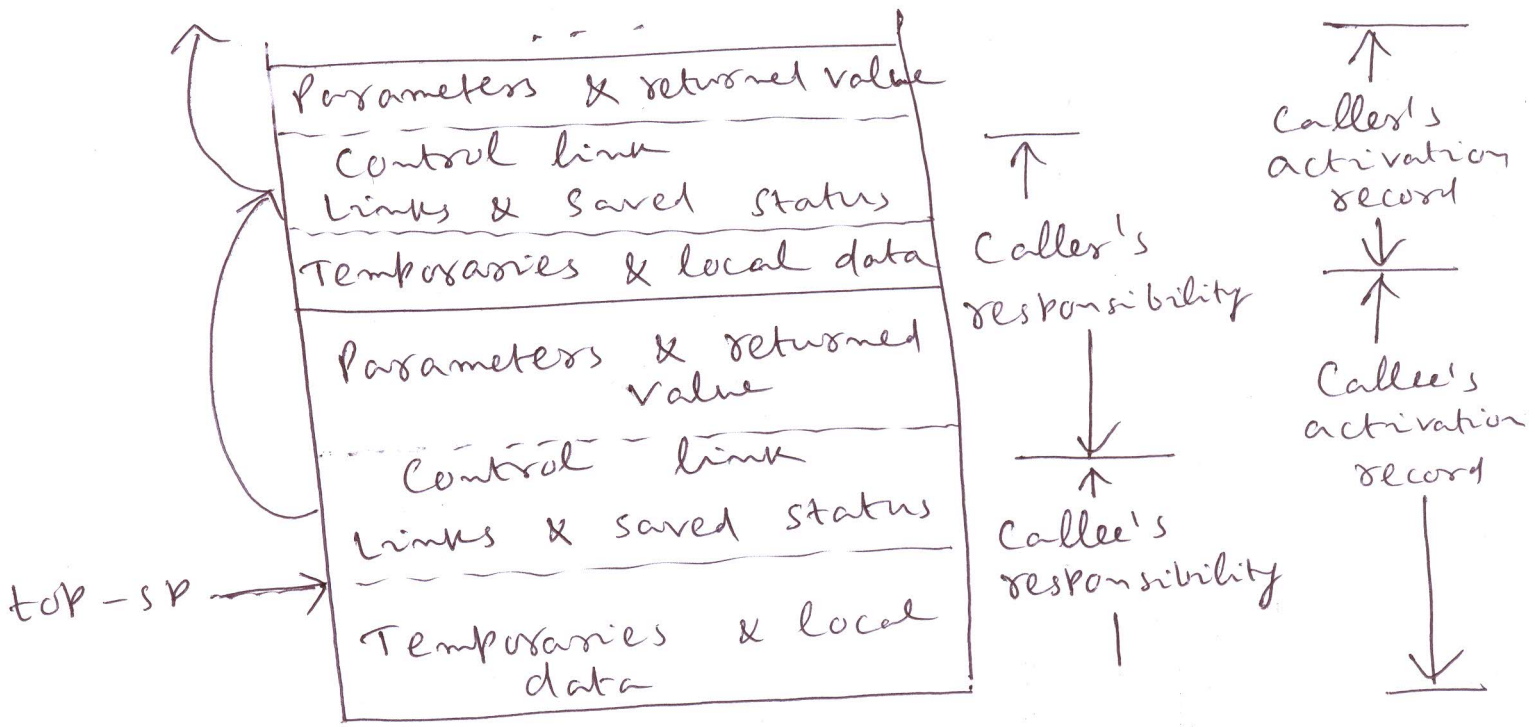
```
int global;
int f()
{
  int i;
  i = 1;
  global = 1;
  global = 2;
  return;
} global = 3;
```

Optimized code

```
int global;
int f()
{
  global = 2;
  return;
}
```

7. Calling Sequence (5+5)M

→ Procedure calls are implemented by what are known as calling sequences, which consists of code that allocates an activation record on the stack and enters information into its fields.



Actions performed during Function call

① The caller evaluates the actual parameters.

② The caller stores a return address & the old value of top-sp into the callee's activation record. Then caller increments top-sp.

③ The callee saves the register values & other status information.

5.(b) Strategy for reducing fragmentation in heap memory (3)

(5M)

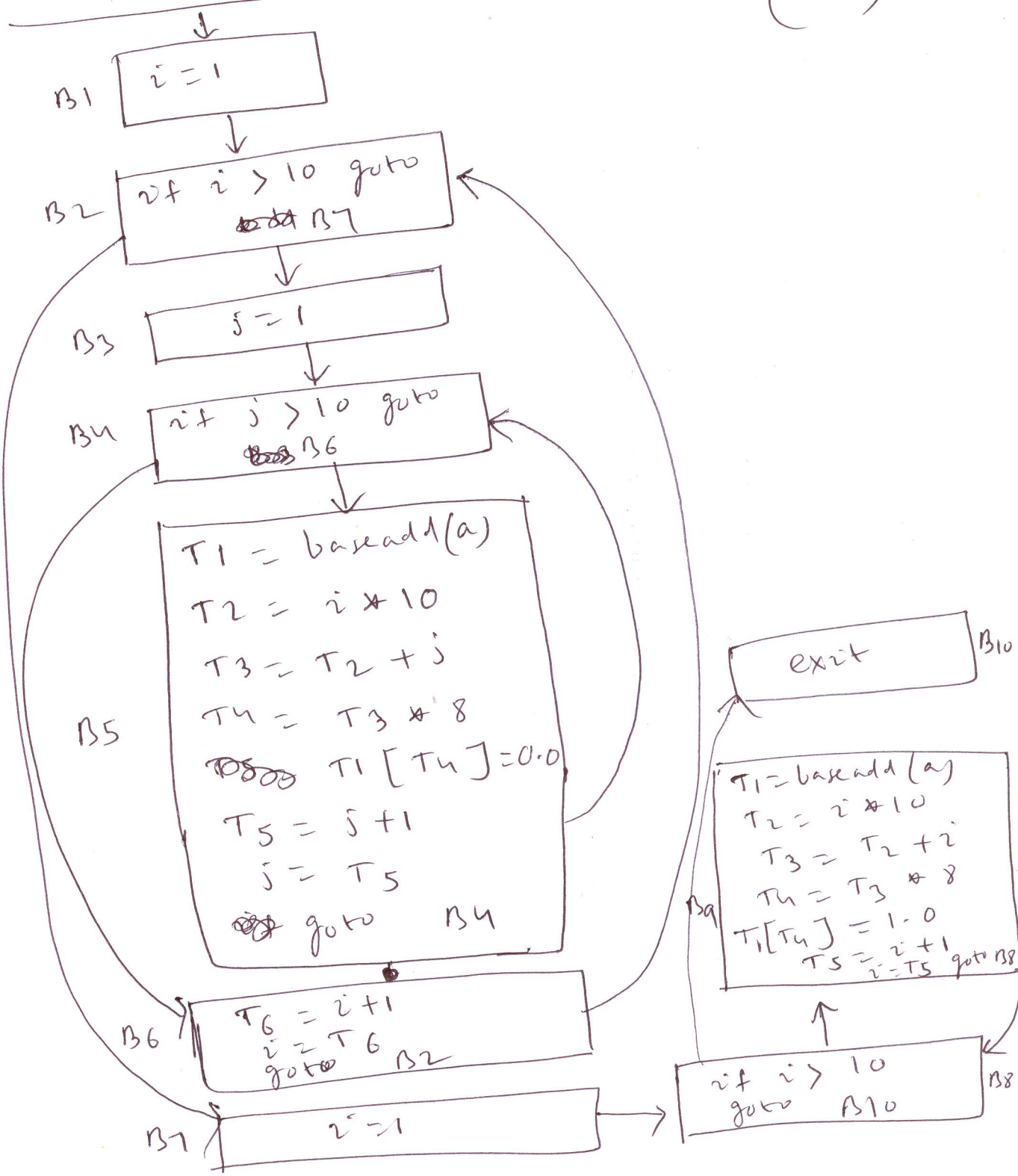
Explanation of

→ Best fit

→ Next fit

6. Intermediate Code

(5+5)M



(4) The callee initializes its local data & begins execution.

Actions performed during Return

(1) The callee places the return value next to the parameters.

(2) Using information in the machine-status field, the callee restores top-sp & other registers & then branches to the return address that the caller placed in the status field.

(3)