Shri Ramdeobaba College of Engineering and Management, Nagpur
Department of Computer Science and Engineering

Sub: Language Processor **Sem: VII Semester**

Practice Problems

Unit-1

Q1. Draw the transition diagram to recognise some of the given keywords of JAVA: case, catch, char, class, const, continue, final, finally, float, for, this, throw, throws, transient.

Q2. Draw the state transition diagram to accept string literals.

Q3. Write a regular expression for accepting date and also draw the transition diagram. Consider the format of date as dd/mm/yy. Days should not extend beyond 31 and only 12 months must be allowed.

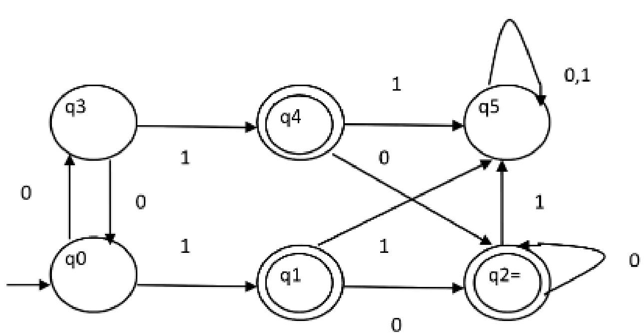
Q4. Write a regular expression to search all the strings that contain the substring "cop".

Q5. Convert the regular expressions to DFA using the direct method: ab(a|b)b

Q6. The number of tokens in the following C statement is:__

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

Q7. Find the minimum DFA for:



UNIT-2

Q1. Show the working of backtracking parser using the given grammar:

S→ (L) | a
L→L,S | S

For the string (a,(a,a))

Q2. Find whether the given grammar is LL(1) or not.

S→ AB | PQx
A→xy | m
B→bC
C→bC | ε
P→pP | ε
Q→qQ | ε

Q3. Construct the LL(1) parsing table for the given grammar

A→aCDq | aBg | ε
C→ p | Ct | BD | rAb | ε
D→ d | ε
B→ e | ε

Q4. Consider the given grammar to find handle and viable prefixes for the input string "((a,a),a)"

S→ a | ^ | (T)
T → T,S | S

Q5. Consider the ambiguous grammar

S→ AS | b
A→SA | a

(i) Construct the collection of sets of LR(0) items for the grammar
(ii) Construct the DFA and Parsing table using SLR algorithm

Q6. Construct SLR parsing table and parse the input string "ijnnj" to show the presence of shift-reduce conflict.

X→ iXYj | jY
Y→kY | mX | Z
Z→Zn | n

Q7. Construct the CLR parsing table for the given grammar:

S → id = E
E → E * E
E → (E)
E → E + E
E → id

Q8. Construct the LALR parsing table for Q6.

Q9. Any ambiguous grammar fails to be LR. Show that the given ambiguous grammar is not SLR and show how we can handle ambiguous grammars.

E→ E+E| E*E |id

Q10. Perform phrase level error recovery for SLR table without conflicts constructed in Q9. Show error recovery for string- “id+*id”

Q11. State true or false:

1. SLR is more powerful than LALR
2. CLR is more powerful than SLR
3. CLR produces largest tables
4. CLR is LR(0)
5. LALR(1) may perform a few reductions after the error has been encountered

UNIT-3

Q1. Represent the expression $Z = (-A * B) + (C * D) + (E - F)/(C * D)$ in quadruple, triple and indirect triple notation.

Q2. Find the three address code for the given code:

```
if (a>b and b<c) then
  begin
      x=x+1;
      y=x+2;
  end
else
       x=x+2;
```

Q3. Find the three address code for the given code:

```
while(x>y) do
begin
  a=b+c;
  z=a+z;
end
```

Q4. Generate the parse tree and give the TAC for the given code fragment:

```
for(i=1; i<50;i+1)
if (i<10) then
a=b+1
else
a=c+1
```

Q5. Generate the parse tree and give the TAC for the given code fragment:

```
repeat
begin
i = i+1;
x = y+z;
end
until x < n ;
```

Q6. Translate the following code fragment into intermediate code. Show the parse tree:

```
switch (i + j)
{
case1: x = y + z
default: P = q + r
case2: u = v + w
}
```

Q7. Generate TAC using SDTS:

```
do
if A = 0 then
```

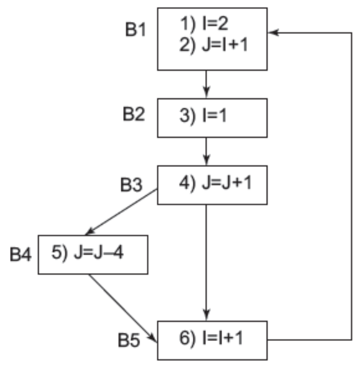
```
A = B+C*D
else
repeat
A = A+1
until A<5
i=i+1;
while(i<10)
```

Q8. Directly write the TAC for
A[I,J + 1]:= B[I,C[I,J]] + D[I,J+1] *E[I,J*2], where w = 4 and
the size of arrays A, B, C, D and E is 10 × 20, 10 × 5, 5 × 5, 10 × 5 and 5 × 20, respectively.

Q9. Translate the following statement into intermediate code:
A[ijk]=B[ij]+C[I + J K]
where
A is 3D array of size 10*10*10
B is 2d 10*10
C is 1d 30

UNIT-4

Q1. Find the data flow equations for the given program flow graph



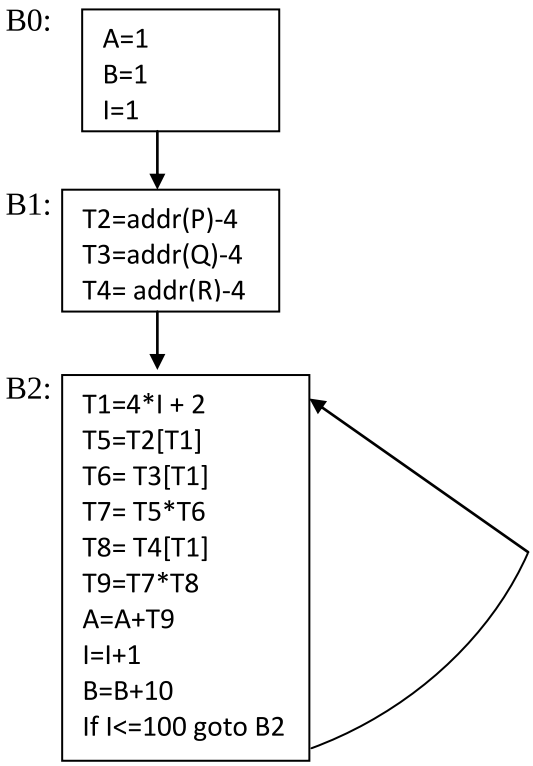
Q2. Find the reaching definitions for the following:

```
i=m−1;
j=n;
a= U1;
do
i=i+1;
j=j-1;
if e1 then
a=U2
else
i=U3
while e2
```

Q3. Find the Program flow graph and detect the loop in the following:

```
a=0;
b=1;
c=2;
L2: if (b>100) goto L3;
a=a+1
d=e+f
L1: if (b>50) goto L3;
c=a;
g=10*d;
h=g+c;
b=b+2;
goto L1;
b=b+4;
goto L2;
L3: i=b;
```

Q4. Perform induction variable elimination



Q5. Perform code optimization for the given code using appropriate code optimization techniques.

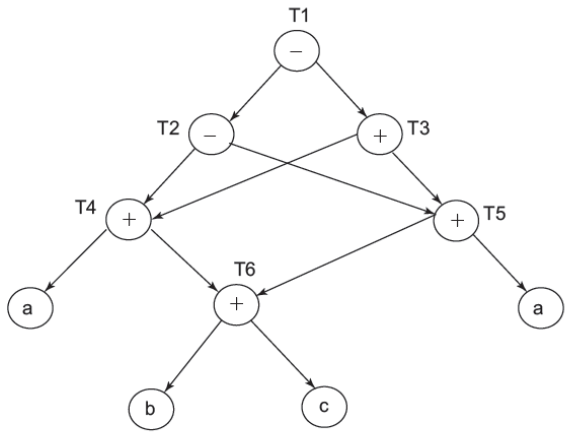
```
B=5
A=B+6
C= D+E
P=C
E=Z+C
F=A+E
```

UNIT-5 (Code Generation)

Q1. Consider the TAC given and generate code using simple code generation algorithm:

x=a[i]
y=b[i]
z=x*y

Q2. Generate Code using getreg() for the given DAG.



Q3. Apply dynamic programing algorithm to find the cost vector and generated code:

g= a*(b+c) + d * (e-f)

Assuming there are 2 registers- R0 and R1

Q4. Generate the optimal order of execution using heuristic algorithm and then generate code using simple code generation

T1=x*y
T2=z+x
T3=w/T2
T4=T1+T3

Q5. Construct the DAG and find the number of registers needed.

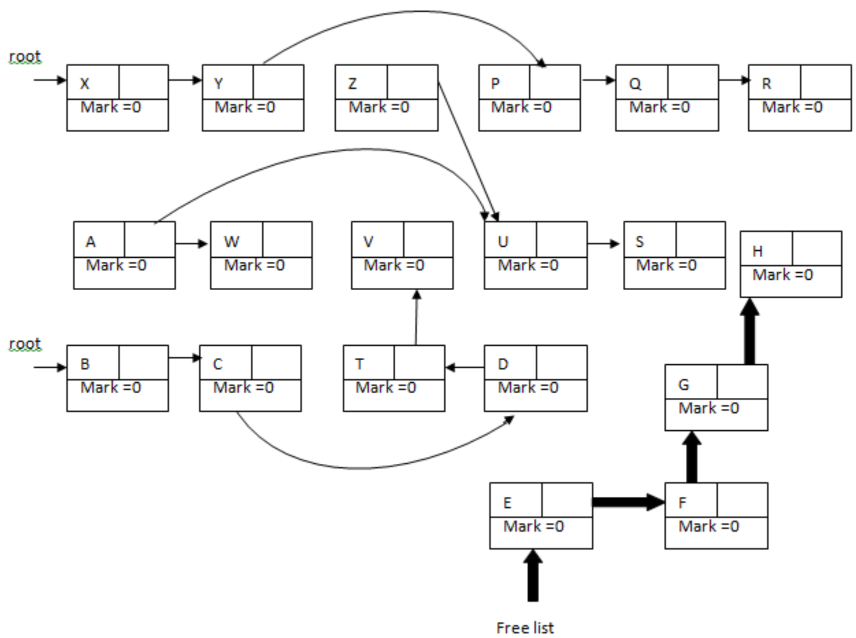
a+a *(b-c) + (b-c) *d

Q6. Consider the three-address code corresponding to expression as given below:

p = a - b
q = a - c
r = q
s = r + p

UNIT-6

Q1. Write the Mark and sweep algorithm and apply it to the given memory. Show contents after mark and after sweep phase



Q2. Apply free space allocation methods to the given memory blocks. Shaded blocks are free blocks. Show the blocks that will be utilised in different strategies if memory request is for a block of size 21.

10	20	50	7	8	100	35	60	40	15	13	7	47	78	7	8	25

Q3. Consider the code fragment and build the symbol table in tree organization and in table with nesting depth.

```
main()
{
      int x,y;
      int A( ){

               char R( ) {
                          int c;
                          char a,b;
                          }
              }
      int B( ){
               real s;
               int p,q;

               int C( )
               {
               int s,t;
                       int D( )
                       {
                       int r;
                       }
               }
               }
}
```

Q4. Write the C code for merge sort. Draw the activation tree when numbers 5 8 1 9 4 2 7 3 are to be sorted. Also show the intermediate control stacks having the activation records.

Q5. Justify how hash table can be used as a data structure to store symbol table. What will the complexity for performing a search operation?