DHARMSINH DESAI UNIVERSITY, NADIAD
FACULTY OF TECHNOLOGY
B.TECH. SEMESTER VI [INFORMATION TECHNOLOGY]
SUBJECT: (IT 608) LANGUAGE TRANSLATOR

Examination	**: First Sessional**	**Seat No.**	**:**
Date	**: 11/01/2018**	**Day**	**: Monday**
Time	**: 12:00 to 1:15**	**Max. Marks**	**: 36**

INSTRUCTIONS:

1. Figures to the right indicate maximum marks for that question.
2. The symbols used carry their usual meanings.
3. Assume suitable data, if required & mention them clearly.
4. Draw neat sketches wherever necessary.
5. Here | is rule separator and ^ stands for NULL.

Q.1 Do as directed.

(a) Discuss: Is there any compilation error in below statement? Also show output generated. **[2]**

printf("%d")

(b) Why regular language is required to make a token? Justify with proper example. **[2]**

(c) Apply code optimization for given C code. **[2]**

```
1. int foo(a, b)
  {
     a = a - b;
     b++;
     a = a * b;
     return a;
  }
```

```
2. for (i = 0; i < 100; )
  {
     do_stuff(i); i++;
     do_stuff(i); i++;
     do_stuff(i); i++;
     do_stuff(i); i++;
     do_stuff(i); i++;
     do_stuff(i); i++;
  }
```

(d) What is lexical, syntax and semantic types of error? Identify these errors in given source code in 'C-language' below (Q.1 (e)). Discuss in brief. **[3]**

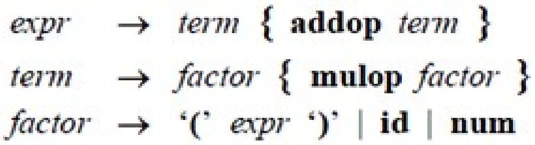
(e) Generate token, lexeme, and pattern for given code in 'C-language'. **[3]**

```
#include<stdio.h>
long int multiplyNumbers(int n);
int main()
{
   int n, 1n;
   printf("Enter a positive integer: ");
   scanf("%d", &n);
   printf("Factorial of %d = %ld", n, multiplyNumbers(n));
   return 0
}
long int multiplyNumbers(int n)
{
   if (&n >= 1)
      return n*multiplyNumbers(n-1)
   else
      return 1;
}
```

Q.2 Attempt ***Any Two*** from the following questions. **[12]**

(a) Generate Recursive Descent Parser (RDP) for given Grammar G1. **[6]**

Grammar G1:



expr → *term* { **addop** *term* }
term → *factor* { **mulop** *factor* }
factor → '(' *expr* ')' | **id** | **num**

Also give the drawbacks of RDP method.

(b) Write Lex code for 'C Scanner' **with automated tool**. [Minimum six tokens] **[6]**

(c) Write Lex code for 'C Scanner' **without automated tool**. [Minimum six tokens] **[6]**

Q.3 (a) Consider given Grammar G2. **[8]**

Grammar G2: E→ E + E | E - E | E * E | E / E | (E) | id

Justify that given grammar G2 is LL(1) or not?

Generate LL(1) grammar for given CFG.

(b) Apply panic mode error recovery mechanism for input string **") id + / id $"** for Grammar G2 (Q.3 (a)). **[4]**

OR

Q.3 (a) Consider given Grammar G3. **[8]**

Grammar G3: R→ R'|'R | RR | R* | (R) | a | b

Justify that given grammar G3 is LL(1) or not?

Generate LL(1) grammar for given CFG.

(b) Apply panic mode error recovery mechanism for input string **"a | b * | c $"** for grammar G3 (Q.3 (a) or). **[4]**