Department of Physics
University of Rajasthan, Jaipur

First Semester Mid-Term Test 2016

Paper H02: Programming in C

Time allowed: 2 hrs M.M.: 80

Attempt all questions. Answer all questions in separate answer-sheets for corresponding units.

1) Short type questions, all carries same marks: (each carries 4 marks)
   a) Describe the operation of a general language generator.
   b) Distinguish between static and dynamic semantics.
   c) Write grammar for following assignment: `a = b;`
   d) Define a left-recursive grammar rule.
   e) Write lexemes and tokens in given statement:
      `xx = 3 * number + 25;`
   f) What is the function with that a c-program starts, write a c-program which print: "`your name`".

   g) What does the following code print?

```
for( i = 1; i <= 5; i++ ){
  for( j = 1; j <= 3; j++ ){
    for(k = 1; k <= 4; k++) printf("*");} printf("\n");
  printf("%d\n", i); }
```

   h) Write a c-program that reads in two integers and determines and prints if the first is integer multiple of the second.
   i) What do you understand by pragmatics?
   j) Write a c-program which can print following string:
      `-3, 1, 5, 9, 13, 17, 21`

2) Write a grammar for the language consisting of strings that have *n* copies of the letter `a` followed by the same number of copies of the letter `b`, where $n > 0$. For example, the strings `ab`, `aaaabbbb`, and `aaaaaaaabbbbbbbb` are in the language but `a`, `abb`, `ba`, and `aaabb` are not. Draw parse trees for the sentences `aabb` and `aaaabbbb`, as derived in grammar.

(20 Marks)

Your name: Please go on to the next page...

OR

What does a parse tree mean? An unambiguous grammar for expression is written as follows:

```
<assign> → <id> = <expr>
<id> → A | B | C
<expr>  →  <expr> + <term>
         | <term>
<term>  →  <term> * <factor>
         | <factor>
<factor> → ( <expr> )
           | <id>
```

Make a parse tree and write derivation for the sentence A = B + C * A using above grammar.

(20 Marks)

3) Distinguish between the terms fatal error and nonfatal error. Write a C program to do binary calculations (+, -, *, /, which are given in input expression.) with integers. There are 3 operations on 4 numbers. Please don't use `while` in the program. Output should as follows:

(20 Marks)

Input: `2+3+4+5`	Input: `2/3+4*5`	Input: `3/3+4/4`
Output: `14`	Output: `20`	Output: `2`

OR

Write a c program that reads in the side of a square and then prints a hollow square. Your program should work for squares of all side sizes between 1 and 20. For example, if your program reads a size of 5, it should print

```
*****
*   *
*   *
*   *
*****
```

(20 Marks)

End of Examination

Solution of midterm question paper

These answers are not unique. You may have your own way of writing and you may have different way of solving the same problem. These are few of correct answers. Few answers may found as it is in Concepts of Programming Language by Robert B. Sebesta.

---

1) This question have multiple parts. All are as:
   a) A language generator is a device that can be used to generate the sentences of a language. It is like a button and we have to press it for producing a sentence of the language. Since, the particular sentence produced by a generator is unpredictable. Generators can be more easily read and understand instead of recognizers. We can compare the syntax of a particular statement with the structure of the generator.

   b) Static semantic is indirectly related to the meaning of program during run/execution. It is the legal form of program (syntax rather than semantics). Analysis required to check specifications can be done at compile time, so it is known as Static semantics.

      Dynamics semantics gives the meaning of expressions, statements and programming language. Here programmer need to know, what exactly the statements do.

   c) Grammar for assignment `a = b;`

```
<assign> → <id> = <expr>
<id> → A | B
<expr> → <var>
          | <id> ;
```

   d) A left recursive grammar is a grammar in which the non-terminals (say A) will eventually derive a sentential form with itself as the left-symbol. In this process a parser can get into a loop in the parsing rules without making any progress consuming the input.

   e)

Lexemes	**Tokens**
xx	identifier

=	equal_sign
3	int_literal
*	mult_op
number	identifier
+	plus_op
25	int_literal
;	semicolon

Please have a look at page number 115, article 3.2 in Sebesta 10th edition .

f) Starting function in c program is `main()`.

```
/*
  midterm question 1f.
  c program to print "your name"
*/
#include <stdio.h>
int main(){
   printf("your name\n");
   return 0;
}
```

g) Output of given part of program where i, j and k are declared as integers.

```
************
1
************
2
************
3
************
4
************
5
```

h) Read 2 integers and compare them with each other as integer multiple of other.

```
/*
   midterm question 1h
   read 2 integers and compare them as integer multiples
*/
```

```
#include <stdio.h>
int main(){
   int a, b;
   printf( "Give 2 integers: " );
   scanf( "%d%d", &a, &b);
   if(a%b==0)
      printf("%d is integer multiple of %d and we can\n"
          "get it by multiplying by %d\n", a, b, a/b);
    else printf("%d is not integer multiple of %d\n", a,
b);
   return 0;
}
```

i) Progmatics is the third general area of language description, referring to practical aspects of how constructs and features of a language may be used to achieve various objectives.

j) Printing `-3, 1, 5, 9, 13, 17, 21` in c-program.

```
/*
  midterm 1j
  printing given number string
*/
#include <stdio.h>
int main(){
  int i;
  for( i = -3; i < 20; i = i + 4 )
    printf("%d, ", i);
  printf("%d\n", i);
  return 0;
}
```

Another way to write same:

```
#include <stdio.h>
int main(){
  int i;
  for( i = 0; i < 6; i++ )
    printf("%d, ", -3 + i*4);
  printf("%d\n", 21);
  return 0;
}
```

One more way:

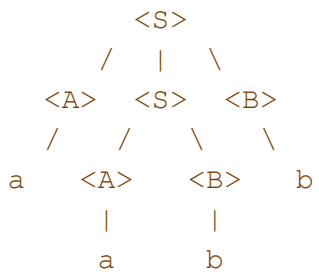
```c
#include <stdio.h>
int main(){
   int i;
   for( i = 0; i < 6; i++ ){
      static int temp = -3;
      printf("%d, ", temp);
      temp = temp + 4;
   }
   printf("%d\n", 21);
   return 0;
}
```

Be careful in this program, examiner is not asking to print comma just after the last integer 21. So, in case you should print it separately.

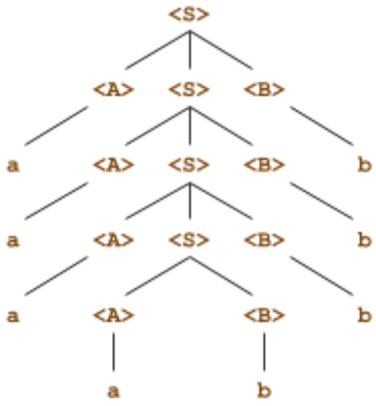
2) Grammar for given strings:

<S> → <A><B> | <A><S><B>
<A> → a
<B> → b

Parse tree for aabb:



Parse tree for aaaabbbb:



OR

While describing features of grammars, they naturally describe the hierarchical syntactic structure of the sentences of the language. These hierarchical structures are called parse trees.

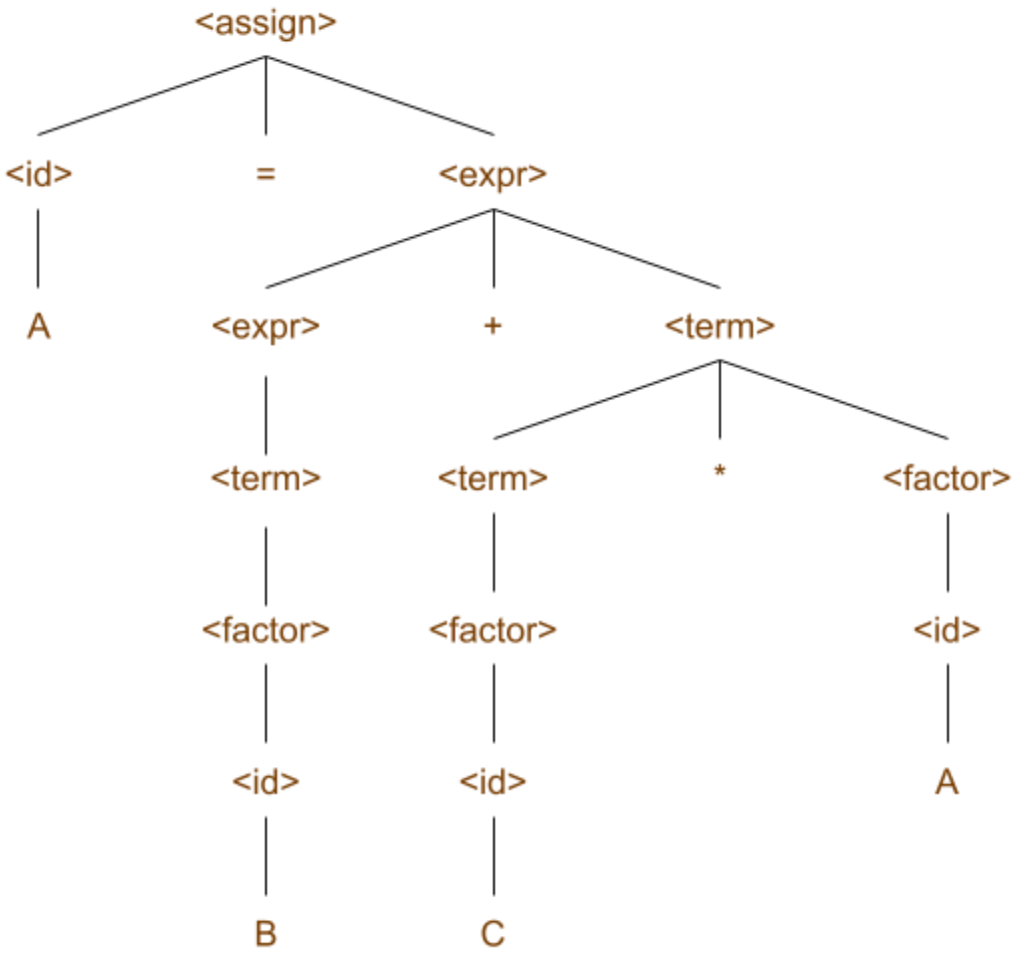
Derivation for the sentence `A = B + C * A` : (leftmost derivation)

```
<assign> => <id> = <expr>
         => A = <expr>
         => A = <expr> + <term>
         => A = <term> + <term>
         => A = <factor> + <term>
         => A = <id> + <term>
         => A = B + <term>
         => A = B + <term> * <factor>
         => A = B + <factor> * <factor>
         => A = B + <id> * <factor>
         => A = B + C * <factor>
         => A = B + C * <id>
         => A = B + C * A
```

Derivation for the sentence `A = B + C * A` : (rightmost derivation)

```
<assign> => <id> = <expr>
         => <id> = <expr> + <term>
         => <id> = <expr> + <term> * <factor>
         => <id> = <expr> + <term> * <id>
         => <id> = <expr> + <term> * A
         => <id> = <expr> + <factor> * A
         => <id> = <expr> + <id> * A
         => <id> = <expr> + C * A
         => <id> = <term> + C * A
         => <id> = <factor> + C * A
         => <id> = <id> + C * A
         => <id> = B + C * A
         => A = B + C * A
```

Unique parse tree for `A = B + C * A` using given grammar is as:



3) An error that causes the program to terminate immediately without having successfully performed its job is known as *fatal error*. Error which allows a program to run to completion, often producing incorrect results is known as *nonfatal error*. If we divide a number with zero (0), then it is not defined in computer and such error is known as *fatal error*. If we have calculated a number such that it is out of it's size range, then program will not give any error message and will give us output. Such errors are known as *nonfatal error*s. A programmer would like to see fatal errors instead of nonfatal error because in case of nonfatal error, we have to look and do many calculations manually.

Program for calculation:

```
/*
  Question 3
  Simple calculations without while
*/
#include <stdio.h>
int check(char);
int operation(int, int, char);
int not1(char);
int main(){
```

```
   int a1, a2, a3, a4;     char c1, c2, c3;  int i, j = 0;
   printf("Give required string: ");
   scanf("%d%c%d%c%d%c%d", &a1, &c1, &a2, &c2, &a3, &c3, &a4);
   if(not1(c1)*not1(c2)*not1(c3) == 0){
      printf("Undefined operation/s is/are used.\n"
           "No result can be derived.\n");
      return 0;
   }
   for(j=0; j<2; j++)
      if(c1 == '/' || c1 == '*'){
         a1 = operation(a1,a2,c1);
         c1=c2; a2=a3; c2=c3; a3=a4;
      } else if(c2 == '/' || c2 == '*'){
         a2 = operation(a2,a3,c2);
         c2= c3; a3 = a4;
      } else {
         a1 = operation(a1,a2,c1);
         c1=c2; a2=a3; c2=c3; a3=a4;
      }
   a1 = operation(a1, a2, c1);
   printf(" = %d\n", a1 );
   return 0;
}
int check(char c){
   if(c == ' ' || c == '\t' || c == '\n') return 1;
   else return 0;
}
int operation(int a, int b, char c){
   if(c=='+') return a+b;
   if(c=='-') return a-b;
   if(c=='*') return a*b;
   if(c=='/') return a/b;
}
int not1(char x){
   if(x == '+' || x=='-' || x=='*' || x=='/') return 1;
   else return 0;
}
```

In above program, scanf statement can be replaced with following:

```
      scanf("%d", &a1);
      for(; ; ){
         scanf("%c", &c1);
         if(check(c1)!=1) break;
      }
      scanf("%d", &a2);
      for(; ; ){
         scanf("%c", &c2);
         if(check(c2)!=1) break;
      }
      scanf("%d", &a3);
      for(; ; ){
         scanf("%c", &c3);
```

```
        if(check(c3)!=1) break;
      }
      scanf("%d", &a4);
```

OR

```
/*
  Question 3 OR:
  Hollow square with stars (*)
*/
#include <stdio.h>
int main() {
   int i, j, n;
   printf( "Enter an integer which say about square size: " );
   scanf( "%d", &n );
   if( n < 0 || n > 20 ) {
      printf( "Given size is not in allowed range.\n" );
      return 0;
   }
   for( i = 1; i <= n; i++ ) {
      for( j = 1; j <= n; j++ )
         if( i==1 || j==1 || i==n || j == n ) printf( "*" );
         else printf( " " );
      printf( "\n" );
   }
   return 0;
}
```

Other way, few students followed:

```
/*
  Question 3 OR:
  Hollow square with stars (*)
*/
#include <stdio.h>
int main (){
   int i, j, n;
   printf( "Enter an integer which say about square size: " );
   scanf( "%d", &n );
   if( n==1 ){
      printf( "*\n" );
      return 0;
   }else if( n>20 ){
      printf( "Size is not allowed more than 20.\n" );
```

```
        return 0;
    }
    for( i = 1; i <= n; i++ ) printf( "*" );
    printf( "\n" );
    for( i = 2; i <= n-1; i++ ) {
        printf("*");
        for( j = 2; j <= n-1; j++ ) {
            printf( " " );
        }
        printf( "*\n" );
    }
    for ( i = 1; i <= n; i++ ) {
        printf( "*" );
    }
    printf( "\n" );
    return 0;
}
```