CD- M2 T1 Question Bank

1. Generate the TAC for
   a. Expression a < b or c < d and e > notf.
   b. The following C Code

```
While(A<C[i] and B>D[i])
{
If A=1 then C[i]++;
Else while A<=D[i] do
  D[i]=D[i]+C[i]
I++
}
```

   c. If[(a<b) and ((c>d) or (a>d))] then

```
 z = x + y * z
Else
 z = z+1
```

   d. C Code for basic calculator by switch statement
   e. C code for scientific calculator by using functions.

2. Consider the piece of code

```
int i
int a [10] [10]
i=0
while (i < 10)
{
a [i] [i] = 1;
i++;
}
```

   a. Design the AST.
   b. *Translate the into three address code.*
   c. *Represent the TAC in various forms*
   d. *Evaluate the address of a[4][1] and a[4][2] by RMO.*

3. Implement the factorial program in C
   a. Annalise the best Intermediate codes.
   b. Represent the TAC in various forms.
   c. Design Activation record for every recursive call in case of factorial (4).

4. Consider switch statement of C language with the syntax as follows:

```
 switch (E)
{
case v1 : S1
case v 2 : S2
…………………..
…………………….
case Vn-1: Sn-1
```

default: Sn
where E is an integer (int) expression, v 1, v 2, ... , Vn-1 are (distinct) integer constants and S 1, S 2, ... , Sn- 1 are list of statements. The semantics of switch statement is defined as in C.

A. Define a suitable grammar for switch statement. Assume vi's as literals. There can be zero or more ca'se clauses and zero or one default clause. For [10+ simplicity you may assume that all case clauses precede the default clause. Note: You do not need to write the production rules forE and S.
B. Propose a translation scheme for switch statement 'based on your grammar. Show the semantic
actions for every production of the grammar. 10+ Note: There is no need to show the actions for E and S as you have not expanded the production rules for them.
C. Generate the 3-address codes for the following sample:
switch (x)
{
case 1: y x; break;
case -1: y -x; break;
default: y 1; break;
Note: Plug in the 'codes for assignment and break statements from your general understanding of 3-address translation as there are no explicit production rules in your grammar generating these statements. Also assume that variables x and y have been declared earlier and hence have been added to an appropriate symbol table. Make further assumptions as needed and document them clearly.

5. Design an intermediate code representation for a new programming language called "LangX." LangX is a statically typed language with support for functions, control structures, and arrays. Discuss the design considerations you would take into account when creating the intermediate code representation for LangX. Explain the data structures and instructions you would use, and provide examples of LangX code and the corresponding intermediate code to illustrate your design with example.

6. Consider any high-level programming language that supports a wide range of operations and data types. As part of the compiler, you need to generate TAC for that program.
   a. Describe how you would use TAC to optimize a specific code snippet in the program to improve its performance.
   b. Provide examples of the original program, the inefficient TAC representation, and the optimized TAC representation. Explain the optimization you applied and its benefits.

7. Imagine you are working on the development of a compiler for a high-level programming language. You have implemented various optimization techniques, including peephole optimization, to enhance the generated code's efficiency.
   a. Explain how peephole optimization can improve the performance of the compiled code. Provide specific examples of code patterns or situations where peephole optimization can be highly effective.
   b. The programming language includes constructs like loops, conditionals, and function calls. The target architecture is a modern, pipelined, and out-of-order execution processor. You can assume that the code generated by the compiler is in assembly language or a low-level intermediate representation (IR).
   c. Discuss both the benefits and limitations of peephole optimization in the context of code generation for this architecture. In your response, consider aspects like code size reduction, instruction reordering, and the elimination of redundant operations.

8. Develop a quick sort code in C programming language,
   a. Develop the intermediate code
   b. Construct a flow graph that represents the control flow of the program. Include basic blocks, control flow edges, and any relevant control flow structures such as loops.
   c. Implement basic block scheduling and register allocation to optimize the generated code.
   d. Generates assembly code from the flow graph.

9.
   a. Identify and remove unreachable code blocks, taking into account control flow and dependencies in the DAG.
   b. Given a DAG representing a high-level code, create an algorithm to generate optimized assembly code from the DAG. Consider various code generation strategies and choose the one that minimizes the number of instructions and register allocation.

10.
   a. How can data flow analysis techniques (constant folding, dead code elimination, and common subexpression elimination) be applied to optimize a Fibonacci program written in C?
   B. Develop the optimized machine code for a basic block within the Fibonacci program?

___________________________________---