Universidade Católica Dom Bosco
Curso de Engenharia de Computação
Disciplina: Compiladores II
Professor: Ricardo Santos
RA: **Nome:** **Data:**

Exame Final

1) (valor 2,0)Assinale a alternativa correta (V ou F)

( ) Dada a expressão: $\texttt{result} := x + (x + 1) * (x + 1) - (x + 1) * y$, a representação na forma de DAGs possui menos vértices que a representação através de uma árvore sintática. Exatamente, essa redução é de 6 vértices devido ao reuso da sub-expressão $x + 1$.

( ) Exemplos de ações semânticas são: verificação de tipos e inserção de símbolos da linguagem na tabela de símbolos.

( ) A otimização "Identidade Algébrica" possibilita transformar uma expressão do tipo: $x = y + 0$ em $x = y$.

( ) No algoritmo de Alcance de Definições ("Reaching Definitions"), o conjunto $\texttt{out}$ para um bloco básico B pode ser definido como: $\texttt{out}_B = \texttt{in}_B \bigcup (\texttt{gen}_B - \texttt{kill}_B)$.

( ) Um GFC (ou CFG) representa apenas o fluxo de controle de um programa através de seus blocos básicos.

( ) Uma das razões para a adoção de mais de uma linguagem intermediária por um compilador está na possibilidade de explorar a geração de código para mais de uma linguagem de máquina.

( ) Sobre a organização da memória em tempo de execução, as variáveis globais de um programa são armazenadas no segmento de dados, as variáveis locais de procedimento são armazenadas no segmento de pilha do registro de ativação do procedimento e os dados alocados dinamicamente ficam no segmento de heap.

( ) Atributos herdados são sempre definidos após à execução de uma regra de produção enquanto que atributos sintetizados são executados em meio à execução da regra.

( ) A geração de código de três endereços para um comando `break`, poderia ser traduzida como um `goto` para um label fora da estrutura de controle `S`. Para tanto, esse `goto` deve ser adicionado para um ponteiro `S.nextlist` antes da redução de `S`. Nesse caso, considere que `S` é a estrutura de controle que contém a instrução `break`.

( ) Dado o seguinte código:

```
int a; float b; a%b
```

O compilador deve informar um erro detectado através da verificação dinâmica de tipos já que o operador módulo (%) não aceita operandos do tipo float.

2) (valor 1,0)Diferencie os algoritmos de *garbage collection*: Contagem de Referências e Marcar e Varrer. Você pode utilizar exemplos para detalhar sua resposta.

3) (valor 1,5)Gere código de três endereços para o corpo do procedimento a seguir. Obs: Não esqueça que o acesso a todo elemento de um vetor do tipo `int` deve estar alinhado em 4 bytes.

```
void insertion_sort_rec(int *vetInt, int n){
 int i, j, elemI;

 if ( n > 1 ) {
   insertion_sort_rec(vetInt, n-1);
   elemI = vetInt[n-1];
   j = n - 1;

   while ( j >= 0 && ( vetInt[j-1] > elemI)){
     vetInt[j] = vetInt[j - 1];
     j--;
   }
   vetInt[j] = elemI;
 }
}
```

4) (valor 2,0)Baseando-se no código a seguir, indique quais otimizações poderiam ser realizadas sobre o código a seguir e mostre o código final após a aplicação de cada otimização. Procure justificar por que a otimização pode (ou não) ser realizada sobre o código.

```
s=0
i=0
L1:
if i<n goto L2
j=4*i
k=j+a
x=M[k]
s=s+x
i=i+1
goto L1
L2:
s=s/2
i=i-i
```

5) (valor 2,0)Considere o programa P a seguir e utilize o algoritmo de list scheduling para definir um escalonamento com menor quantidade de ciclos possível. Considere que a operação de multiplicação consome 2 ciclos e as demais operações consomem apenas 1 ciclo. Informe o Delay de cada operação, e o conteúdo dos conjuntos Cand, MCands, ECand e Sched após cada passo do algoritmo.

```
p=e+2
f=2*a
c=d>>1
g=b*c
j=f*g
m=j-f
i=p<<2
t=m<<i
```

6) (valor 2,0)Mostre a árvore de ativação junto com a pilha de registros de ativação (considere o valor do parâmetro, o valor de retorno, o acesso às variáveis globais e as variáveis locais) para o programa a seguir. Obs: O conteúdo do procedimento insertion_sort é apresentado no exercício 2.

```
int main(void){
  int vetInt[] = { 11, 8, 10, 9, 3 };
  insertion_sort( vetInt, 5);
}
```