University of Toronto

CSC488S/CSC2107S Compilers & Interpreters — **Winter 2002**
Midterm Test [15% of final mark] — **March 5, 2002**

Total marks 100 - Total time is 50 minutes. Two pages, 5 questions. Answer all questions.
Instructions: This midterm is open book, open notes. Non-programmable calculators allowed. No electronic communication devices allowed.
The line numbers in example programs and grammars are for reference only and are not a part of the programs or grammars.

1. [20 marks] Lexical analysis for the $Java^{TM}$ programming language must process Unicode escape sequences which are defined as

```
\ UnicodeMarker hexDigit hexDigit hexDigit hexDigit
```

Where `UnicodeMarker` is one or more instances of the letter `u` and hexDigit is any one of the characters `0123456789abcdefABCDEF` . Because the \ character has other uses in $Java^{TM}$ a \ character is the start of a Unicode escape sequence if and only if it is immediately preceeded by an even (possibly zero) number of other \ characters. If it is immediately preceeded by an odd number of \ characters it is not the start of a Unicode escape sequence. There must be exactly four hexDigits in a valid Unicode escape sequence. Examples:

Input	Processed	Comments
`\u2297`	$\otimes$	Valid Unicode Escape for the character $\otimes$
`\\u2297`	`\\u2297`	Odd number of preceeding \s
`\\\uuu005A`	`\\Z`	Valid Unicode escape for the character Z
`\ \u2260`	\ $\neq$	Valid unicode escape for the character $\neq$ (there is a space after \ )
`\udefg`		Error, not 4 hex digits

Unicode escape sequences are processed before the main part of lexical analysis.
Describe a lexical analysis algorithm for processing Unicode escape sequences in $Java^{TM}$.
Describe any interactions with other parts of lexical analysis. Discuss error handling.

2. [25 marks] Show how the data structure (Z) declared below would be laid out in memory using the fill minimizing structure allocation Algorithm 2 as described in lecture.
Give complete details of the layout showing the offsets of all fields.

```
union {
        struct{ char A ; int B ; char C ; double D ;  } X ;
        struct{ short P ; char Q ; double R ; int S ; } Y ;
} Z[2] ;
```

You should assume the length and alignment factors for atomic data types shown in the table below.

Type	length	align	Type	length	align
char	8	8	int	32	32
short	16	16	double	64	64

3. [20 marks] Describe the static semantic analysis checks that a C compiler would make on the fragments of C code shown below.

```
int I, J = 7, A[ N ] ;
char *S, T[128] = "Hello World" ;
...
S = malloc( sSize );
strncpy( S, T, I - J );
for( J = 0 ; J <  sSize ; J ++ )
    if( S[ J ] != A[ J ] )
        A[ J ] = S[ J ] ;
printf("The answer is %d (%s)\n", I + J , S );
```

4. [15 marks] In the lectures it was recommended that the symbol table entries for minor scopes (i.e. embedded scopes delimited by $\{$ and $\}$) should be merged with the symbol table of the closest enclosing major (i.e function or procedure) scope. Describe a *complete* symbol table algorithm for implementing minor scope merging. Discuss

- What happens at the start of a minor scope
- What happens at the end of a minor scope
- How the symbol table lookup algorithm is modified to deal with minor scopes.

5.[20 marks] Convert the grammar given below to an LL(1) grammar that defines the same language. λ is the empty string.

1	S	=	's' ,
2			'b' D 'm' L 'e'
3	L	=	F L 'm' S ,
4			S
5	D	=	'd' 'm' D ,
6			'd'
7	F	=	'f' ,
8			λ

List the LL(1) director sets for your revised grammar.