Vidyavardhini College of Engineering & Technology, Mumbai, India

MCA (SEM – I)

C Programming

Assignment No.1

Q. 1 Write short notes on:

I. Keywords and Identifiers

Keywrords: Keywords are preserved words that have special meaning in C language. The meaning of C language keywords has already been described to the C compiler. These meaning cannot be changed. Thus, keywords cannot be used as variable names because that would try to change the existing meaning of the keyword, which is not allowed.(Don't worry if you do not know what variables are, you will soon understand.) There are total 32 keywords in C language.

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
const	extern	return	union
char	float	short	unsigned
continue	for	signed	volatile
default	goto	sizeof	void
do	if	static	while

Identifiers: In C language identifiers are the names given to variables, constants, functions and user-define data. These identifier are defined against a set of rules.

Pre-processor commands

Rules for an Identifier:

1. An Identifier can only have alphanumeric characters (a-z , A-Z , 0-9) and underscore(_).
2. The first character of an identifier can only contain alphabet (a-z , A-Z) or underscore (_).
3. Identifiers are also case sensitive in C. For example **name** and **Name** are two different identifiers in C.
4. Keywords are not allowed to be used as Identifiers.
5. No special characters, such as semicolon, period, whitespaces, slash or comma are permitted to be used in or as Identifier.

When we declare a variable or any function in C language program, to use it we must provide a name to it, which identified it throughout the program, for example:

```
int myvariable = "Studytonight";
```

Here `myvariable` is the name or identifier for the variable which stores the value "Studytonight" in it.

II. getchar() and gets()

getchar(): The C library function **int getchar(void)** gets a character (an unsigned char) from stdin. This is equivalent to **getc** with stdin as its argument.

Following is the declaration for getchar() function.

- `int getchar(void)`

This function returns the character read as an unsigned char cast to an int or EOF on end of file or error.

The following example shows the usage of getchar() function.

```
#include <stdio.h>
```

```
int main () {
   char c;

   printf("Enter character: ");
   c = getchar();

   printf("Character entered: ");
   putchar(c);

   return(0);
}
```

gets(): The C library function **char *gets(char *str)** reads a line from stdin and stores it into the string pointed to by str. It stops when either the newline character is read or when the end-of-file is reached, whichever comes first.

Following is the declaration for gets() function.

- char *gets(char *str)

Parameters:

str − This is the pointer to an array of chars where the C string is stored.

This function returns **str** on success, and **NULL** on error or when end of file occurs, while no characters have been read.

Eg.,

The following example shows the usage of gets() function.

```
#include <stdio.h>

int main () {
   char str[50];

   printf("Enter a string : ");
   gets(str);

   printf("You entered: %s", str);

   return(0);
}
```

III. putchar() and puts

IV. printf() and scanf()