EXTERNAL PAPER QUESTIONS

Compiler and Interpreter

Difference between Compiler and Interpreter

No	Compiler	Interpreter
1	Compiler Takes **Entire** program as input	Interpreter Takes **Single** instruction as input .
2	Intermediate Object Code is **Generated**	**No** Intermediate Object Code is **Generated**
3	Conditional Control Statements are Executes **faster**	Conditional Control Statements are Executes **slower**
4	**Memory Requirement : More** (Since Object Code is Generated)	**Memory Requirement** is **Less**
5	Program need not be **compiled** every time	Every time higher level program is converted into lower level program
6	**Errors** are displayed after **entire program** is checked	**Errors** are displayed for **every instruction** interpreted (if any)
7	**Example** : C Compiler	**Example** : BASIC

Break and Continue

break	continue
A `break` can appear in both `switch` and loop (`for`, `while`, `do`) statements.	A `continue` can appear only in loop (`for`, `while`, `do`) statements.
A `break` causes the `switch` or loop statements to terminate the moment it is executed. Loop or `switch` ends abruptly when break is encountered.	A `continue` doesn't terminate the loop, it causes the loop to go to the next iteration. All iterations of the loop are executed even if `continue` is encountered. The `continue` statement is used to skip statements in the loop that appear after the `continue`.
The `break` statement can be used in both `switch` and loop statements.	The `continue` statement can appear only in loops. You will get an error if this appears in switch statement.
When a `break` statement is encountered, it terminates the block and gets the control out of the `switch` or loop.	When a `continue` statement is encountered, it gets the control to the next iteration of the loop.
A `break` causes the innermost enclosing loop or `switch` to be exited immediately.	A `continue` inside a loop nested within a `switch` causes the next loop iteration.

Example:
for(i=1;i<10;i++)
{
if(i%5==0)
break;
else
printf("%d",i);
}
Output:
1234
	Example:
	for(i=1;i<10;i++)
	{
	if(i%5==0)
	continue;
	else
	printf("%d",i);
	}
	Output:
	12346789

getch() and getchar()

getch()	getchar()
Getch is used to hold the output sceen and wait until user gives any type of input(i.e. Until user press any key)	This function is used to take input of a single character.
Syntax: int getch(void);	Syntax: var name=getchar();

Example:
void main()
{
clrscr();
printf("%c",getch());
getch();
}
	Example:
	void main()
	{
	char c;
	clrscr();
	c=getchar();
	printf("You have entered %c",c);
	getch();

Pre Decrement and Post Decrement

Pre Decrement	Post Decrement
First value is decremented, then assign to variable.	First value assign to variable, then decremented.
Syntax : --var_name;	Syntax : var_name--;

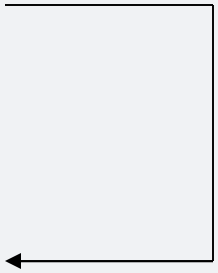
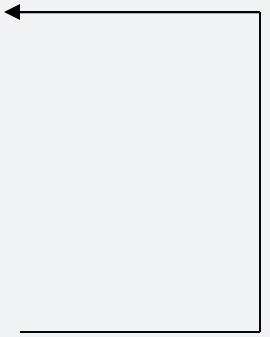
Example:
void main()
{
int a=10;
printf("%d",--a);
getch();
}
Output:
9
	Example:
	void main()
	{
	int a=10;
	printf("%d",a--);
	getch();
	}
	Output:
	10

= and ==

=	==
= is Assignment Operator	== is Comparison or Relational Operator.
It is used to assign a value to variable.	It is used to compare two values.

Example:
void main()
{
int a=10; // Assignment Operator
printf("%d",a);
getch();
}
	Example:
	void main()
	{
	int a=10,b=20;
	if(a==b) // Relational Operator
	printf("Equal");
	else
	printf("Not Equal");
	getch();
	}

Forward Jumping and Backward Jumping

Forward Jumping	Backward Jumping
One or more statements are skipped and program continue at later stage.	One or more statements are repeated and program continue at above stage.
Syntax: goto label; ---- ---- ---- lable : Statements; 	Syntax: label; Statements; ---- ---- ---- goto label; 

Example:
void main()
{
goto a;
printf("VNSGU");
a:
printf("SURAT");
getch();
}
	Example:
	void main()
	{
	a:
	printf("VNSGU");
	printf("SURAT");
	goto a;
	getch();
	}