①

ID	14525
Name	Manahil Khan
Subject	Programming fundamental
Program	BS(CS)
Teacher name	DR. Fazal-e-Malik
Date	29/9/20

ID:
14535

QNO # 1

Part (a)

* Purpose of if Statements:-

If Statements in C++ is used to control the program flow based on some condition. its used to execute some Statements code block if the expressions is evaluated to true. Otherwise it will get skipped. This is the simplest way to modify the flow of the program. The if Statement in C++ can be used in various forms depending on the situation & complexity.

Two different forms with example

```
// TWO types of if condition are if-THEN
   & IF - THEN ELSE
// IF Condition
#include <iostream>
 main()
{
   if(condition)
 {
// Body of if Statement
```

```
    }
}
```

```
// IF - THEN_ELSE condition
main()
{
   if(condition)
  {
   // Block of code if condition is true
  }
 else{
     // block of code if condition is false
      }
}
```

OR

a) if (conditions)
   do this;

(b) if (condition)
   do this;
   else
   do this;

Id: 14525

Part(b) (4)

```
int main()
{
    int n1, n2;
    cout<<"Enter two numbers from keyboard:\n";
    cout<<" enter first number\n";
    cin>>n1;
    cout<<" enter 2nd number\n";
    cin>>n2;
    if(n1>=n2)
    {
     cout<<"Largest numbers is:";
     cout<<n1;
    }
    else{
        cout<<"Largest number is:";
        cout<<n2;
        }
        return 0;
}
```

Id: 14525

(5)

QNO#2

Part(a)

logical operators:-

logical operators are used to combine two or more conditions/constraints or to complement the evaluation of the orginal conditions in consideration. The result of the operation of a logical operator is a boolean value either true or false

examples:-

Operator	Description	Example
&&	called logical AND operator if both the operands are non-zero, then condition becomes true.	(A&&B) is false
||	Called logical OR operator if any of the two operands is non-zero, then condition becomes true.	(A||B) is true

Id:
14525

!

Q6

Called logical
not operator
use to reseverses
the logical state
of its operand if
a condition is
true, then logical
not operator will
make false

!(A&&B) is
true

Id:
14525

Part (b) ⑥⑦

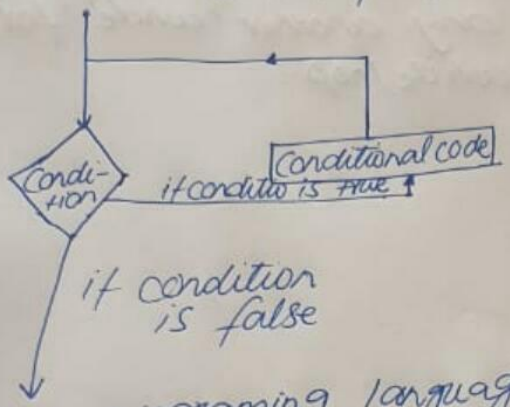
```
#include<iostream>
#include<conio.h>
using namespace std;
main()
{
    int tmp;
  cout<<" temperature is\n";
  cin>>tmp;
 if(tmp>=40)
 {
  cout<<" its very hot\n";
 }
 else if(tmp>35 && tmp<40)
 {
  cout<<" its tolerable\n";
 }
else if(tmp>=30 && tmp<=35)
 {
  cout<<" its warm\n";
 }
else
 { cout<<"cool";
 }
}
```

Id:
14535

QNO#3

Part(a)

A loop statement allows us to execute a statement or group of statements multiple times & following is the general from of a loop statement in the most of the programing languages



C++ programing languages provide the following type of loops to handle looping requirements

* loops types:-

1) While loops:

Repeats a statement or group of statements while a given condition is true. it tests the condition before executing the loop body.

Id: 14525

2) for loop:-

Execute a sequence of statements multiple times & abbreviates the code that manage the loop variable

3) do-while loop:-

like a while-statement, except that it test the condition at the end of the loop body.

4) nested loops:-

you can use one or more loop inside any another 'while' 'for' or 'do' —— while loop.

Id:
14525

(10)

Part (b)

```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{
   int number;
  cout<<"Enter the number\n";
  cin >> number;
  if (number %.2 == 0)
    cout << number << " is an even number";
else
  cout << number << " is an odd number";
  return 0;
}
```

Id: 14535

(14)

QNO#4

Part (a)

* Purpose of using Break & Continue Statements:-
* The Break Statement
* There are situations where we want to jump out of a loop instantly, without waiting to get back to the conditional test.
* The keyword break allows us to do this.
* When break is encountered inside any loop, control automatically passes to the first statements after the loop.
* A break is usually associated with an if.
* The keyword break, breaks the control only form the loop in which its placed.
* Continue Statements:-
* continue Statement allows to take the control to the begining of the loop. bypassing the statements inside the loop, which have not yet been executed
* A continue is usually associated with an if.

Id:
14525

Part (b)

```
#include<iostream>
#include<conio.h>
using namespace std;
int main()
{
   int i;
   int Sum=0;
  cout<<"The first 10 natural numbers are:\n";
  for(i=1; i<=10; i++)
  {
    cout<<i<<" ";
    Sum=Sum+i;
  }
  cout<<"\n The Sum of first 10 natural numbers is\n";
  cout<<Sum;
  return 0;
}
```

Id:
14525

(13)

QNO# 5

Part (a)

* C++ character set

Alphabates :-

A, B, ... Y, Z

a, b, ... y, z

Digits :-

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

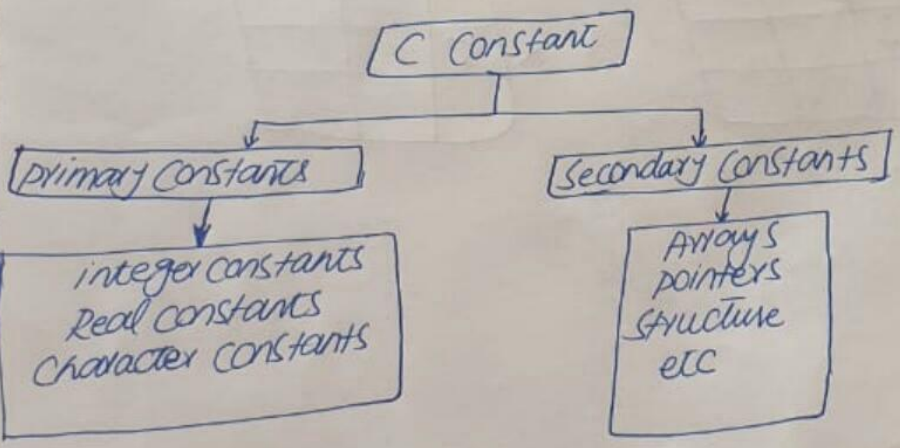
Special Symbols :-

~ ! # $ % ^ & * ( ) _ - = + { } [ ] ;

: " ' < > ? , . / | \ '

Part (b)

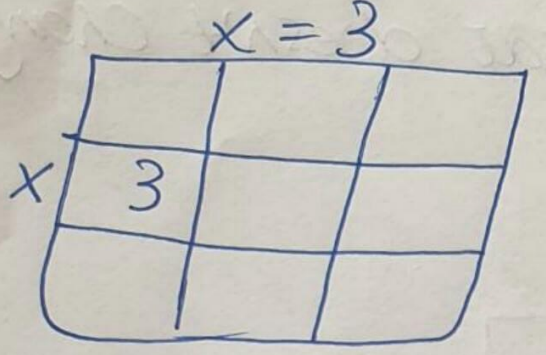
* Constants is an entity that doesn't change.

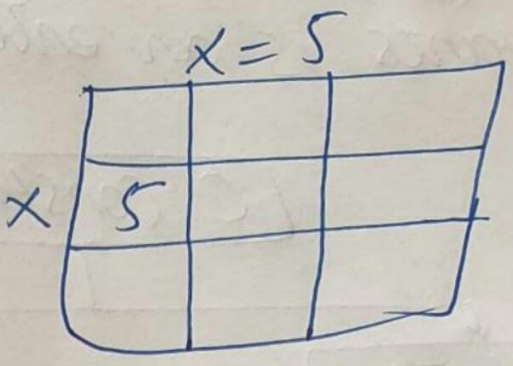


Part(c)

Variables:-

* An entity that may change during program execution.
* These are names given to locations in memory.





Id: 14525

(15)

part (D)

Keywords

* These are reserved words
* Compiler know their meaning
* Cannot be used as variable name
* Cannot be changed

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

part(e)

~~Relational operators~~

* Relational Operators:-

Standard algebric Relational operator	C++equality	Example	Meaning
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than & equal to y

Id: 14525

≤	<=	x<=y	x is less than or equal to y

Equality operator

=	==	x==y	x is equal to y
≠	!=	x!=y	x is not equal to y