Name Iftikhar khan

ID 14693

class BS (SE) B

Paper Programming Fundamental.

Teacher Name Dr. Fazal-e-Malik.

Q=1 (a)

Ans = 1 (a)

- ELSE-if statement:

The else-if statement is useful when you need to check multiple conditions within the program, nesting of if-else blocks can be avoided using else-if statement.

Syntax of else-if statement:

```
if (condition)
{
    // these statements would execute if the
       condition 1 is true
}
else if (condition 2)
{
    // these statements would execute if
       the condition 2 is true
}
    else if (condition 3)
{
    // These statements would execute if
       the condition 3 is true.
```

}

: else

{

// These statement would execute if all condition return false.

}

* ═══ * ═══ * ═══ *

Q = 1 (b)

Ans = 1 (b)

```
#include<iostream>
main(){
    int a = 10;
    int b = b;
    if (a > b)
    {
        cout<<"a is the largest numeber";
    }
    else
    {
        cout<<" smallest";
    }
}
```

Q=2 (a)

Ans 2 (a)

- Logical Operators:-

A logical operators is a symbol or word used to connect two or more expressions such that the value of the compound expression produced depends only on that of the original expressions and on the meaning of the operator. Common logical operation include AND, OR and NOT.

Example:-

```
// c program to demonstrate working of relational operators.
#include <stdio.h>
int main()
{
    int a = 10, b = 4;
    // greater than example.
    if (a > b)
        printf("a is greater than b\n");
else
    Printf ("a is less then or equal to b\n");
```

(4)

(2)+

// greater than equal to

```
if (a>=b)
    printf("a is greater than or equal
                to b/n");
    printf ("a is lesser than b/n");
// not equal to
    if (a!=b)
    printf ("a is not equal to b/n");
    else
    printf ("a is equal b/n");
    return 0;
}
```

= * = * = * =

Q = 2 (b)

Ans = 2 (b)

```
#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 (temp>35 && tmp<40)
    {
    cout<<" its tolerable \n");
    }
    else if (tmp>=30 && tem<=35)
    {
    cout<<" its warm \n";
    }
    else
    {
        cout<<" cool ".
```

Q = 3 (a)

Ans = 3 (a)

· Looping :-

A loop statement allows us to execute a statement or group of statement multiple times and following is the general from of a loop statement in most of the programming language c++ programming language provides the following type of loops to handle looping requirements.

Two types of loops :-

1) Entry controlled loops. in this type of loops the test condition is tested before entering the loop body. For loop and while loop are entry controlled loops.

2) Exit controlled loops: In this type of loops the test condition is tested or evaluated at the end of loop body.

Q = Q3 (b)

Ans = Q3(b)

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

Q = 4 = (a)

Ans (a)

- Break Statements::

~ * ~ when a Break statements is executed, the most deeply nested loop currently being exected is ended and execution picks up with the next statement after the loop. For example, consider the following gprogramming:

```
while (1) {
    if (n < 0) break;
    foo (n);
    n = n - 1;
    }
```

- continue statements::

~ * ~ The continue statements ends the current operation of the loop and returns to the condition at the top of the loop. such loops are typically used to exclude same values from calculations. For example, we could.

(8) (9)

use the following loop to sum the positive values in the array x;

```
real sum;
sum = 0;
for (n in 1: size (x)) {
if (x[n] <= 0) continue;
sum += x[n];
}
```

Q = 4 (b)

Ans 4 (B)

C++ program 1+2+3 ··· 10.

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

* = * = * = *

Q5 = (a)

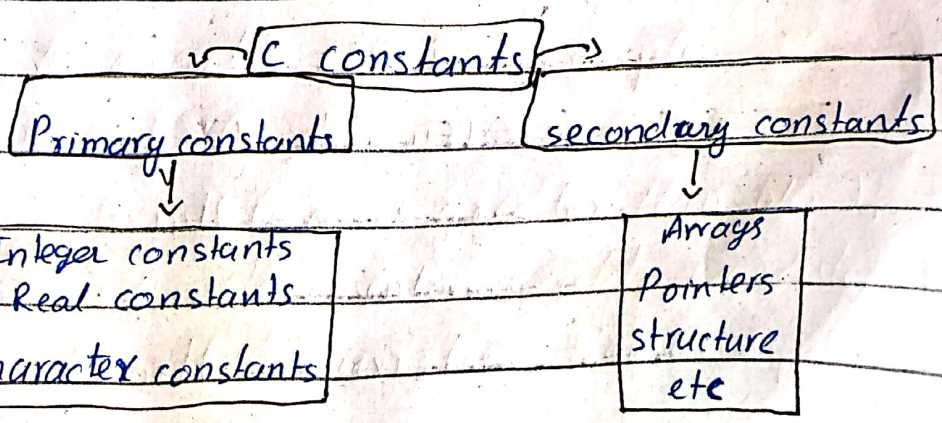
Ans 5 (a)

(a) c++ character set:

Alphabets	A, B, --- Y, Z
	a, b, --- y. z
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
Special	~ ! # $ % ^ & * ( ) _ - = +
Symbols	{ } [ ] ; : " ' < > ? , . / | \

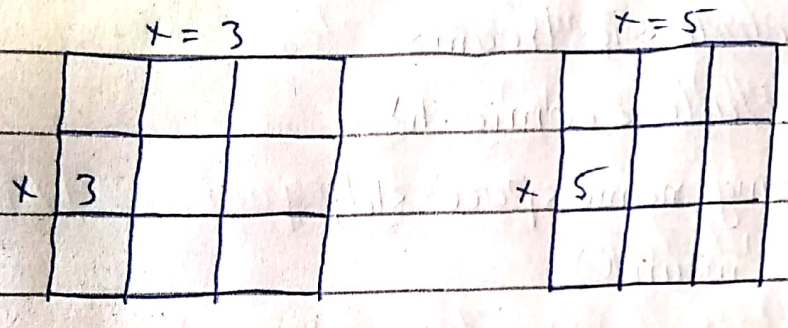
(b) constants:

constant is an entity that doesn't change



c) Variables:

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



- series of characters (letters, digits, underscores)
- Must begin with a letter or underscore
- case sensitive.
- Meaningful naming scheme:

d) Key words:

- These are reserved words.
- compiler knows their meaning.
- cannot be used as variable name
- Cannot be changed.

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef.

e) Relational Operators:

Standard algebraic	c++ equality	Example	Meaning.
Relational operator			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than
≤	<=	x <= y	x is less
Equality operator			
=	==	x == y.	x is equal
≠	!=	x != y	x is not equal.