Q1:

i) Arr < Employee> workerlist (100);

This is a template function Arr is template.
Employee is class used by a template and class 100.
when defining class template function for from a class template, it is important to provide a type to complete the definition of the new type being declared. eng

=> This statement is used to store data for any data type. The datatype is define by user in compiletime.

ii)

If we use template for same datatype the same static data member is used. If we use different type then different static member are used.

So number of static members are depend on number of datatype for which it is called.

Moreover static data members of class-template must be declared at file scope.

e.g

Template Z :: i

Z is static member of class while the statement in the main () function assigns the value.

iii)

static members can used with out object of class created by using class name (::) scope resolution operator and name of static member.

class name :: static member function();

iv)

A copy constructor is needed when we pass an object of class to a function as parameter even when dynamic memory is not involved!

```
class [redacted] A
{ public :
        int x, y :
        A ( A & obj )
        {
        x = obj x ;
        y = obj y ;
        }
void print ( ) { coot << x << y ; }
};
```

v)

we should use const wherever possible so that objects are not accidentally modified. This is one good reason for passing reference as const but there is more if a compiler create a temporary object which is copied by using copy constructor in original program.

—.

void func (A, obj) { }

vi) when we use pointer or dynamic memory in class then constructor should be made. If we use default constructor Then compiler give garbage value (address) to pointer then pointer allocation to garbage memory which will created runtime error.

If memory deallocated property it causes memory leak.

Q2

```
#include <iostream>
#include <string>

class student
{
    int rollNo;
    public:
        student()
        {
            cout<<" student Constructor"<<endl;
        }
void getNum()
    { cout<<"Input Roll No"; cin>>rollNo;
        }
void putNum()
    {
    cout<<" Roll Number" <<rollNo;
    }
};
```

```
class test: virtual public student
{ public:
    int A, B;
    void getmarks() { cin>>A>>B; }
    void putmarks() { cout<<A<<B<<endl;
                    }
};

class sports: public virtual student {
   public :
        int score;
void getscore() { cin>> score; }
void putscore() { cout<< score; }
};
```

```
class result: public test, public sports
{
    int total;
  public:
void output () {

total = A + B + Score;
 put numbers();
  put marks();
   put score();
    cout << total;
        }
};

int main () {
  Student  * std = new std;
   std → getnumbers();

student  * std1 = new test;
    std1 → get marks();

student * std2 = new sports;
    ptr 2 → get score();
```

```
std *p= std3= new result;
    std 3 ->output();
system("pause>0");
}
```

Q2

b)

~~class Another~~

line 28 will transfer tehe ~~control~~ line 25 line 26 will run.

lin 27 will transfer to control to line 28.

c)

Firstly the objects are not declared. ~~the~~
Secondly objects are not calling fonctions defined in above classes.

class Another but doesnot have the function display On Screen. So the guess (ai) wont work.
Secondly guess doesnot deal with nulls.

```
#include <iostream>
#include <string>
#include <string.h>
using namespace std;

template <class var>
class Calculator {
    var value_1;
    var value_2;
public:
    Calculator() {}
    Calculator(var a, var b) //var a, var, b
    {
        value_1 = a;
        value_2 = b;
    }
    var Addition();
    var Subtraction();
    var Multiplication();
    var Dirision();
    var Modulus();
};
```

```
template < class var > var Calculator
                                <var>::
                                Addition
{ return value_1 + value_2; }

template <class var> var Calculator <var>:: Subtraction
{ return value_2 - value_1; }

template < class var > var Calculator <var>::Multiplication
{ return value_2 * value_1 ; }

template < class var> var Calculator <var>::
                                Division
{ return (value_1 / value_2) ; }

template < class var > var Calculator <var>::
                                Modulus()
{ var temp= static_cast<int>(value1) %
            static_cast<int>(value_2);
    return temp;
}
```

```
int main()
{
    int choice;
    string input1;
    string input2;

    Calculator obj_C;

try {
    cout << "Enter 1st Number" << endl;
    cin >> input1;

for ( int i=0; i< input1.size(); j++)
{
    if (! is digit (input1[j]))
    {
        throw input1;
    }
}

}
catch (string exception)
{
    cout << "Invalid input :-( " << endl;
```

```
cout << "Enter 1st Number : ";
while (true) {
start1:
    cin >> input1;
for (int i=0 ; i< input1.size(); j++)
  {
    if (!is digit (input1[i])) {
    cout <<" Input Number only "<<e
      goto    start1;
    }
}
    break;
}
}

try
  {
  cout <<" Enter 2nd number : ";
    cin >>  input2;
for (int i=0; i<input2.size(); j++) {
    if (!isdigit (input2 [i]))
    {
      throw  input2;
    }
```

```
catch (string x) {
    cout<< "Invalid input" <<endl;
    cout<< " Enter 2nd Number : ";
    while (true) {
    start2 :
        cin>> input2;

for (int i=0; i< input2.size(); i++)
    { if (! isdigit (input2[i])) {
        Cout<<"Incorrect Number"<<endl;
        goto start2;
    }
    } break;
}
    double Num1 = stod (input1);
    double Num2 = stod (input2);

Calculator<double>obj_C (Num1, Num2);
```

```
cout << "Addition : " << obj.Addition()<<endl;
cout << " Subtraction : " << obj_C.Subtraction()
                          <<endl;

cout<< "Multiplication " << obj_C.Multiplication()
                         << endl;

cout << " Division " << obj_C.Division()
                    <<endl;

cout << " Modulus " << obj_C.Modulus()
                    << endl;

return 0;
 system ("pause>0");
```

Q5

```
#include <iostream>
using namespace std;

class Counter
{
  int count;
public:
  Counter (int c=0) : count(c) {}

void operator ++() { ++count; }

Counter Counter :: operator ++(int n)

{   count ++;
    return *this;
}

friend Counter operator ++ (int n)
{
    count ++;
    return *this;
}
```

```
friend operator+ ( counter obj1,
                   counter obj2);

friend ostream &operator << (ostream
                  &output, counter &obj)
};

ostream & operator << (ostream &output,
  counter & obj) {
    out << " count " << obj.count <<
    endl;
    return output;
}

counter operator + (counter obj, int n)
{
    obj.count = obj.count + n;
}
```

```
Counter operator +=( counter obj1,
                     counter obj2)
{
    obj1.count t= obj2.count;

    return obj1;
}

int main()

{ Counter c(10), b;

cout<< c++ << endl;

b+ - ++c + c++;

b = b+12;

cout << b << endl;

system("pause>0");
```

Q4: Hospital Managment

```
#include <iostream>
#include <string>

using namespace std;

class hospital; // forward declaration
class [illegible] Store.
class
{ protected;

    int   ID;
    char  name[30];
    int   age;
    char  data[30];
    char  data1[30];
public:
 void  setID(int id)
 { ID= id; }

 void  setName(char *nam)
 {

    char *A;
    A = nam;
For (int i=0;i<
```

```
{ name[i] = A[i]; }

void setAge (int a) { age = a; }
int retAge () { return age; }

int retID() { retusnID; }
char * setName()
   { retuon name() }

void setData (char *d)
   {  char * A;
      A=d;
      for (int i=0; i<30; i++)
   { data[i] = A[i]; }
   }

void setData1 (chas *d1)
   {  Chas *A; A=d1;
   for (int i=0; i<30; i++)
      { data1[i] = A[i]; }
   }
```