Ans1:

Source code:

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

using namespace std;

class Computer{
   protected:
      string processors;
      string companyName;
      int num_cores;
   public:
      Computer(string cN,string p, int cores){ // constructor
         companyName = cN;
         processors = p;
         num_cores = cores;
      }

      virtual void get_detail(){ // virtual function to get details
         cout<<"\nCompany Name: "+ companyName<<"\nProcessor: "+processors<<"\nNumber of Cores: "+num_cores<<endl;
      }
};

class PersonalComputer: public Computer{
   protected:
      string type;
   public:
```

```
    PersonalComputer(string t,string cN,string p, int cores) : Computer(cN,p,cores){ // constructor
        type=t;
    }
    void get_detail(){ // function with same name as in derived class to get details
        cout<<"\nCompany Name: "+ companyName<<"\nProcessor: "+processors<<"\nNumber of Cores: "+num_cores<<"\nType of Computer: "+type<<endl;
    }

};

int main(){

  Computer * c = new Computer("DELL","Intel i5",4); // initialize the computer class
  c->get_detail(); // calling from Computer class

  c = new PersonalComputer("Tablet","Lenovo","Qualcom SnapDragon 855", 8);

  c->get_detail(); // calling from PersonalComputer class
  return 0;
}
```

Output:

```
PS D:\sem-5\oop_Lab\s2> cd "d:\sem-5\oop_Lab\s2\" ; if ($?) { g++ ans1.cpp -o ans1 } ; if ($?) { .\ans1 }

Company Name: DELL
Processor: Intel i5ber of Cores:

Company Name: Lenovo
Processor: Qualcom SnapDragon 855of Cores:
Type of Computer: Tablet
```

Answer 2:

Source Code:

```
#include <iostream>
#include <string>
using namespace std;
class Employee{
private:
string name;
string id;
int salary;
public:
Employee(string n, string i, int s){
id=i;
name=n;
salary = s;
}
virtual ~Employee(){
cout<<"Deleting Employee\n";
}
void getEmployee(){
cout<<"Employee id: "<<id<<"\nEmployee Name: "<<name<<"\nEmployee salary: "<<salary<<endl;
}
};
class Scientist: protected Employee{
private:
int num_publications;
int num_awards;
string *publications;
string *awards;
public:
```

```
Scientist(int num_pub, string *pub, string *aw,int n_a, string name, string id, int
salary):Employee(name,id,salary){
num_publications = num_pub;
num_awards = n_a;
publications = new string[num_publications];
for(int i=0;i<num_publications;i++){
publications[i] =pub[i];
}
awards = new string[n_a];
for(int i=0;i<n_a;i++){
awards[i] =aw[i];
}
}
~Scientist(){
cout<<"\nDeleting Scientist\n";
delete[] publications;
delete[] awards;
}
void getScientist(){
cout<<"\nScientist Details:\n";
getEmployee();
cout<<"\nTotal publications: "<<num_publications<<endl;
cout<<"Publications:\n";
for(int i=0;i<num_publications;i++){
cout<<" "<<publications[i]<<endl;
}
cout<<"\nTotal Awards: "<<num_awards<<endl;
cout<<"Awards:\n";
for(int i=0;i<num_awards;i++){
```

```
cout<<" "<<awards[i]<<endl;
}cout<<"\n";
}
};
class Manager : protected Employee{
private:
string title;
int yrs_of_exp;
int teams;
public:
Manager(string t, int yrs, int tm,string name, string id, int salary):Employee(name,id,salary){
title=t;
yrs_of_exp=yrs;
teams = tm;
}
~Manager(){
cout<<"\nDeleting Manager\n";
}
void getManager(){
cout<<"\nManager Details: \n";
getEmployee();
cout<<"Title as Manager: "<<title<<endl;
cout<<"Years of Experience: "<<yrs_of_exp<<endl;
cout<<"Teams Managed: "<<teams<<endl;
}
};
class Laborer : protected Employee{
private:
int overtime;
```

```
int wage_over;
int leaves;
public:
Laborer(int o, int w_o, int l,string name, string id, int salary) :Employee(name,id,salary){
overtime = o;
wage_over = w_o;
leaves = l;
}
~Laborer(){
cout<<"\nDeleting Laborer\n";
}
void getLaborer(){
cout<<"\nLabourer Details: \n"<<endl;
getEmployee();
cout<<"Overtime: "<<overtime<<endl;
cout<<"Wage in overtime: "<<wage_over<<endl;
cout<<"Total leaves: "<<leaves<<endl;
}
};
int main(){
string a1[]={"alpha", "beta"}, p1[]={"one","two"},a2[]={"gamma", "delta"},p2[]={"three","four"};
Scientist s1(2, p1,a1,2,"Paras","100",100);
Scientist s2(2, p2,a2,2,"Naman","103",10010);
s1.getScientist();
s2.getScientist();
Manager m1("General Manager",6, 10, "Tanmay Vig", "101", 10000000);
Manager m2("Assistant Manager",3, 6, "Manuj Monga", "104", 1000000);
m1.getManager();
m2.getManager();
```

```
Laborer l1(8,500,1,"Rishabh","102",10000);

Laborer l2(10,10,2,"Almas","105",-1000);

l1.getLaborer();

l2.getLaborer();

return 0;

}
```

Output:

```
PS D:\sem-5\oop_Lab\s2> cd "d:\sem-5\oop_Lab\s2\" ; if ($?) { g++ ans2.cpp -o ans2 } ; i

Scientist Details:
Employee id: 100
Employee Name: Paras
Employee salary: 100

Total publications: 2
Publications:
 one
 two

Total Awards: 2
Awards:
 alpha
 beta


Scientist Details:
Employee id: 103
Employee Name: Naman
Employee salary: 10010

Total publications: 2
Publications:
 three
 four
```

```
Total Awards: 2
Awards:
 gamma
 delta


Manager Details:
Employee id: 101
Employee Name: Tanmay Vig
Employee salary: 10000000
Title as Manager: General Manager
Years of Experience: 6
Teams Managed: 10

Manager Details:
Employee id: 104
Employee Name: Manuj Monga
Employee salary: 1000000
Title as Manager: Assistant Manager
Years of Experience: 3
Teams Managed: 6

Labourer Details:

Employee id: 102
Employee Name: Rishabh
```

```
Employee id: 102
Employee Name: Rishabh
Employee salary: 10000
Overtime: 8
Wage in overtime: 500
Total leaves: 1

Labourer Details:

Employee id: 105
Employee Name: Almas
Employee salary: -1000
Overtime: 10
Wage in overtime: 10

Deleting Scientist
Deleting Employee

Deleting Scientist
Deleting Employee
PS D:\sem-5\oop_Lab\s2>
```

Answer 3:

Source code:

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


template <class T>
class Node{
   public:

      T data;
```

```
    Node<T> *next; // points to next node

    Node(T d)
    {
       data = d;
       next = NULL;
    }

};

template <class T>
class Queue{

   Node<T> *start;
   Node<T> *end;

   public:

    Queue()
    {
       start = end = NULL;
    }

    bool empty() // checks if Queue is empty and returns true if yes
    {
       return start==NULL;
    }

    void push(T v) // pushing element at the end of the Queue
```

```
{
   Node<T> *temp = new Node<T>(v);
   if(empty())
   {
      start = end = temp;
   }
   else
   {
      end->next = temp;
      end = temp;
   }

}

T face() // returns element at first position
{
   if(empty())
      return NULL;
   else
      return start->data;
}



void pop()// removes the first element of the Queue
{
   if(empty())
   {
      cout<<"Queue is Empty"<<endl;
   }
```

```
        else if(start==end)
        {
            delete start;
            start = end = NULL;
        }
        else
        {
            Node<T> *temp = start;
            start = start->next;
            delete temp;
        }
    }


};
int main()
{
    Queue<string> q;
    if(q.empty()) cout<<"Queue is empty\n";
    else cout<<"Queue have some elements\n";

    q.push("Tanmay Vig");
    q.push("Rishabh");

    cout<<"Queue Front: "<<q.face()<<endl;

    q.push("How");
    q.push("are");
    q.push("you?");
```

```
    if(q.empty()) cout<<"Queue is empty\n";
    else cout<<"Queue have some elements\n";


    while(!q.empty()){
        cout<<q.face()<<endl;
        q.pop();
    }

}
```

Output: