Answer to the question no 1:

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

class Student
{
   public:

   int id;

};

class Node
{
   public:

   Student value;
   Node *next;
    Node(Student n)
       {
          this->value=n;
          this->next=nullptr;
       }

};

class Linkedlist
{
```

```
private:
    Node *head;

public:
    Linkedlist()
    {
        this->head=nullptr;
    }

    void insertAtBeginning(Student value)
    {
        Node *newNode=new Node(value);
        newNode->next=head;
        head=newNode;
    }

    void insertAtMiddle(int key, Student value)
    {
        Node *temp=head;
        while(temp->next!=nullptr)
        {
            if(temp->value.id==key)
            {
                Node *newNode=new Node(value);
                newNode->next=temp->next;
                temp->next=newNode;
                break;
            }
```

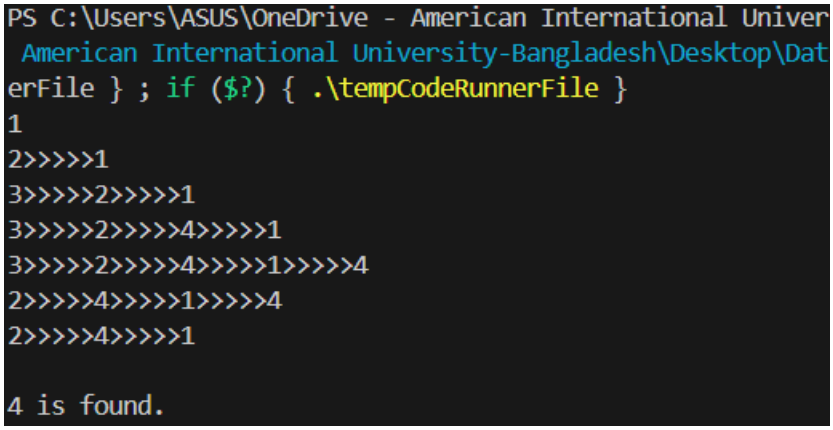
```
        temp=temp->next;
    }
}

void insertAtEnd(Student value)
{
    Node *temp=head;
    while(temp->next!=nullptr)
    {
        temp=temp->next;
    }
    Node *newNode=new Node(value);
    temp->next=newNode;

}

void deleteAtEnd()
{
    Node *temp=head;
    while(temp->next->next!=nullptr)
    {
        temp=temp->next;
    }

    delete temp->next;
    temp->next=nullptr;

}
```

```
void deleteAtBeginning()
{
   if(head!=nullptr)
   {
      Node *temp=head;
      head=head->next;
      delete temp;

   }

   else
   {
      cout<<endl<<"LinkedList is already Empty"<<endl;
   }
}

void search(int key)
{
   Node *temp=head;
   int count=0;
   while(temp->next!=nullptr)
   {
      if(key==temp->value.id)
      {
         cout<<endl<<key<<" is found."<<endl;
         return;
      }
```

```
        else
        {
            count++;
        }
        temp=temp->next;
    }

    if(temp->value.id==key)
    {
        cout<<endl<<key<<" is found."<<endl;
        return;
    }

    if(count>0)
    {
        cout<<endl<<key<<" is not found."<<endl;
    }
}

void printlist()
{
    Node *temp=head;
    while(temp->next!=nullptr)
    {
        cout<<temp->value.id<<">>>>>";
        temp=temp->next;
    }
```

```
            cout<<temp->value.id;

        }

};

int main()
{
    Student s1,s2,s3,s4,s5;
    s1.id=01;
    s2.id=02;
    s3.id=03;
    s4.id=04;
    s5.id =05;

    Linkedlist l1;
    l1.insertAtBeginning(s1);
    l1.printlist();
    cout<<endl;

    l1.insertAtBeginning(s2);
    l1.printlist();
    cout<<endl;



    l1.insertAtBeginning(s3);
    l1.printlist();
```

```
    cout<<endl;

    l1.insertAtMiddle(2,s4);
    l1.printlist();
    cout<<endl;


    l1.insertAtEnd(s4);
    l1.printlist();
    cout<<endl;

    l1.deleteAtBeginning();
    l1.printlist();
    cout<<endl;

    l1.deleteAtEnd();
    l1.printlist();
    cout<<endl;

    l1.search(4);



}
```



Answer to question no 2:

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

class Student
{
   public:

   int id;

};
```

```
class Node
{
   public:

   Student value;
   Node *next;
    Node(Student n)
      {
         this->value=n;
         this->next=nullptr;
      }

};

class Linkedlist
{
   private:
      Node *head;

   public:
      Linkedlist()
      {
         this->head=nullptr;
      }

      void insertAtBeginning(Student value)
      {
         Node *newNode=new Node(value);
```

```
    newNode->next=head;
    head=newNode;
}

void insertAtMiddle(int key, Student value)
{
    Node *temp=head;
    while(temp->next!=nullptr)
    {
        if(temp->value.id==key)
        {
            Node *newNode=new Node(value);
            newNode->next=temp->next;
            temp->next=newNode;
            break;
        }
        temp=temp->next;
    }
}

void insertAtEnd(Student value)
{
    Node *temp=head;
    while(temp->next!=nullptr)
    {
        temp=temp->next;
    }
    Node *newNode=new Node(value);
```

```
    temp->next=newNode;

}




void search(int key)
{
    Node *temp=head;
    int count=0;
    while(temp->next!=nullptr)
    {
        if(key==temp->value.id)
        {
            cout<<endl<<key<<" is found."<<endl;
            return;
        }

        else
        {
            count++;
        }
        temp=temp->next;
    }

    if(temp->value.id==key)
    {
```

```
            cout<<endl<<key<<" is found."<<endl;
            return;
        }

        if(count>0)
        {
            cout<<endl<<key<<" is not found."<<endl;
        }
    }

    void printlist()
    {
        Node *temp=head;
        while(temp->next!=nullptr)
        {
            cout<<temp->value.id<<">>>>>";
            temp=temp->next;
        }
        cout<<temp->value.id;

    }

};

int main()
{
    Student s1,s2,s3,s4,s5;
    s1.id=01;
```

```
s2.id=02;
s3.id=03;
s4.id=04;
s5.id =05;

Linkedlist l1;
l1.insertAtBeginning(s1);
l1.printlist();
cout<<endl;

l1.insertAtEnd(s2);
l1.printlist();
cout<<endl;



l1.insertAtEnd(s3);
l1.printlist();
cout<<endl;

l1.insertAtEnd(s4);
l1.printlist();
cout<<endl;

l1.insertAtMiddle(2,s4);
l1.printlist();
cout<<endl;
```

```
    l1.search(2);


}
```

```
PS C:\Users\ASUS\OneDrive - American International Un
y-Bangladesh\Desktop\Data Structure\Lab\Final Term\Ex
1
1>>>>>2
1>>>>>2>>>>>3
1>>>>>2>>>>>3>>>>>4
1>>>>>2>>>>>4>>>>>3>>>>>4

2 is found.
```

Answer to the question no 3:

```
#include<iostream>
using namespace std;
void swap(int *x, int *y)
{
    int temp=*y;
    *y=*x;
```

```
    *x=temp;
}

void printArray(int arr[], int size)
{
    cout<<endl;
    for(int i=0;i<size;i++)
    {
        cout<<arr[i]<<" ";
    }
    cout<<endl;
}
void bubbleSort(int arr[], int size)
{  int a=0,c=0,d=0;
    for(int i=0;i<size;i++)
    {
        for(int j=0;j<size-1;j++)
        {

            if(arr[j]>arr[j+1])
            {
                swap(&arr[j],&arr[j+1]);
                a++;
            }

        }
        c++;
    }
```

```
    cout<<endl<<"Printing the sorted Array."<<endl;
    printArray(arr,size);

    d=c+a;

    cout<<"\nNumber of comparisions = "<<d<<endl;
    cout<<"\nNumber of exchanges = "<<a<<endl;
}
int main()
{


    int size;
    cout<<"Enter the size of the array : ";
    cin>>size;
    int arr[size];
    cout<<"Enter  your elements : ";
    for (int i = 0; i < size; i++)
    {
        cin >>arr[i];
    }

    cout<<"Printing the original Array."<<endl;
    printArray(arr,size);
    bubbleSort(arr,size);

}
```

```
PS C:\Users\ASUS\OneDrive - American Int
 American International University-Bangl
($?) { .\Bubble_sort }
Enter the size of the array : 6
Enter  your elements : 92
82
21
16
18
95
Printing the original Array.

92 82 21 16 18 95

Printing the sorted Array.

16 18 21 82 92 95

Number of comparisions = 15

Number of exchanges = 9
```

Answer to the question no 4:

```
#include<iostream>

using namespace std;

class Node{
public:
```

```
    int value;
    Node *next;

    Node(int d){
    this->value=d;
    this->next=nullptr;

    }
};
class LinkedList{
private:
    Node *head;
public:
    LinkedList(){
    this->head=nullptr;
    }

    void insertAtBeginning(int value){
    Node *newNode=new Node(value);
    newNode->next=head;
    head=newNode;
    }
    void insertAtMiddle(int element,int value){
    Node *temp=head;
    while(temp->next!=nullptr){
        if(temp->value=element){
            Node *newNode =new Node(value);
            newNode->next=temp->next;
```

```
        temp->next=newNode;
        break;
    }
    temp=temp->next;
}
}
void insertAtEnd(int value){
    if (head==nullptr){
        head=new Node(value);
    }
    else{
            Node *temp=head;
    while(temp->next!=nullptr){
          temp=temp->next;
        }
        Node *newNode =new Node(value);
        temp->next=newNode;

    }


}
 void deleteAtEnd(){
            if (head==nullptr){
         cout<<"empty"<<endl;
         return;
    }
    else if(head->next==nullptr){
```

```
        delete head;
        head= nullptr;

    }
    else{
          Node *temp=head;
    while(temp->next->next!=nullptr){
           temp=temp->next;
        }
        delete temp->next;
        temp->next=nullptr;
    }

}

void deleteAtBeginning(){
    if(head!=nullptr){
        Node *temp=head;
        head=head->next;
        delete temp;
    }
    else{
        cout<<endl<<"LinkedList is already empty"<<endl;
    }
}
void search(int key){
Node *temp=head;
int count=0;
```

```
while(temp->next!=nullptr){
  if(key==temp->value){
     cout<<endl<<key<<" is found."<<endl;
     return;

  }
  else{
     count++;
  }
  temp=temp->next;
}

if(temp->value==key){
  cout<<endl<<key<<" is found."<<endl;
  return;
}
if(count>0){
  cout<<endl<<key<< " is not found."<<endl;

}

}

void printList(){
   if(head==nullptr){
     cout<<"Linked list is empty"<<endl;
     return;
   }
```

```
    Node *temp =head;
    while(temp->next!=nullptr){
        cout<<temp->value<<">>>>>";
        temp=temp->next;
    }
    cout<<temp->value<<endl;
    }
};
class Queue{
public :
   LinkedList l;
   void enqueue(int value){
      l.insertAtEnd(value);
   }
   void dequeue(){
      l.deleteAtBeginning();
      }

   void printQueue(){
      cout<<" Linked List is :"<<endl;
       l.printList();
   }
};

int main(){
   Queue q;
   q.enqueue(1);
   q.enqueue(2);
```

```
    q.enqueue(3);
    q.enqueue(4);
    q.enqueue(5);
    q.enqueue(6);



    q.printQueue();
    q.dequeue();
    q.dequeue();

   cout<<"After impleminting Queue ";
    q.printQueue();

    return 0;


}
```

```
PS C:\Users\ASUS\OneDrive - American Internati
tional University-Bangladesh\Desktop\Data Str
 Linked List is :
1>>>>>2>>>>>3>>>>>4>>>>>5>>>>>6
After impleminting Queue  Linked List is :
3>>>>>4>>>>>5>>>>>6
```

Answer to the question no 5:

```
#include<iostream>

using namespace std;

class Node {
public:
    int data;
    Node* next;

    Node(int d) {
        this->data = d;
        this->next = nullptr;
    }
};

class LinkedList {
private:
    Node* head;

public:
    LinkedList() {
        this->head = nullptr;
    }

    void insertAtBeginning(int data) {
        Node* newNode = new Node(data);
        newNode->next = head;
```

```
        head = newNode;
    }

    void deleteAtBeginning() {
        if (head != nullptr) {
            Node* temp = head;
            head = head->next;
            delete temp;
        }
        else {
            cout << "Linked list is already empty" << endl;
        }
    }

    void printList() {
        if (head == nullptr) {
            cout << "Linked list is empty" << endl;
            return;
        }
        Node* temp = head;
        while (temp->next != nullptr) {
            cout << temp->data << "============>";
            temp = temp->next;
        }
        cout << temp->data;
    }
};
```

```
class Stack {
public:
    LinkedList l;

    void push(int data) {
        l.insertAtBeginning(data);
    }

    void pop() {
        l.deleteAtBeginning();
    }

    void printStack() {
        cout << "Stack: ";
        l.printList();
        cout << endl;
    }
};

int main() {
    Stack s;
    s.push(1);
    s.push(2);
    s.push(3);
    s.push(4);

    s.printStack();
```

```
    s.pop();
    cout << "After popping, ";
    s.printStack();

return 0;
}
```

```
PS C:\Users\ASUS\OneDrive - American International Uni
tional University-Bangladesh\Desktop\Data Structure\La
Stack: 4===========>3===========>2===========>1
After popping, Stack: 3===========>2===========>1
```