19- The Linked list class described in the book is a class that manages dynamic memory. as such, it must overload the copy constructor, the constructor, and the assignment operator.

20- Name and describe the two files authored for each class.

Specification file (H)

Implementation file (CPP)

Traditionally, the class declaration is put in a header file of the same name as the class, and the member functions defined outside of the class are put in a .cpp file of the same name as the class.

Putting class definitions in a header file

We can put functions inside header files in order to reuse them in multiple files or even multiple projects. Classes are no different

21- What is abstraction? How do Classes provide abstraction? What is an abstract data type ?

Data abstraction refers to, providing only essential information to the outside world and hiding their background details

Abstraction separates code into interface and implementation. So while designing your component, you must keep interface independent of the implementation so that if you change underlying implementation then interface would remain intact.

templated classes are abstract data types they provide the detail independent of the actual data type in the application

22- What is the purpose of constructors?

A class constructor is a special member function of a class that is executed whenever we create new objects of that class.

A constructor will have exact same name as the class and it does not have any return type at all, not even void. Constructors can be very useful for setting initial values for certain member variables.

Constructors

special functions that are called automatically upon object creation, to initialize data members

cannot be called via programming without constructors, data members have unpredictable values ...unless brace initialized

the default constructor declarations: no parentheses

constructors with parameters declarations need parentheses

the copy constructor

special purpose: …………………………….

cannot use brace initialization with constructors

23- What is the difference between a static array and a dynamic array?

Static arrays are fixed in size. Dynamic arrays are variable-size

What is an advantage of using a static array over a dynamic array?

Fixed arrays have O(1) insertion, deletion, and reading. They are also trivially easy to implement

Lower cost

What is an Advantage of using a dynamic array over a static array?

If you don't know the size needed for an array until runtime, you need to allocate it dynamically.

Dynamic arrays are variable-size, and therefore are more flexible

Which kind of array uses heap memory?

Dynamic arrays

Which kind of array uses stack memory?

Static array

24- In what 3 situations in the copy constructor called?

1- when an existing object is assigned an object of it own class

```
MyClass A,B;

A = new MyClass();

B=A; //copy constructor called
```

2- if a functions receives as argument, passed by value, an object of a class

```
void foo(MyClass a);

foo(a); //copy constructor invoked
```

3- when a function returns (by value) an object of the class

```
MyClass foo ()
  {
    MyClass temp;
    ....
    return temp; //copy constructor called
  }
```

In C++, a Copy Constructor may be called in following cases

1. When an object of the class is returned by value.

2. When an object of the class is passed (to a function) by value as an argument.

3. When an object is constructed based on another object of the same class.

4. When compiler generates a temporary object.

25- write definition for the default constructor for the Donkey class

```
class A
{
Private:
```

```
string name;

Public:

A(string n=”Fred”):name(n){}

string  getName()const;

{return name;}

};

class  Donkey

{

Private:

A callme;

int  age;

public:

Donkey();  //initialize

// clame.name to “don”, age to 4.

};
```

26- Compare and contrast a linked list and a dynamic array. Compare their memory usage and time usage. Which is better? Name an abstract data type that is better implemented with an array. Name an abstract data type that is better implemented with a linked list.

Why Is This Linked List Better?

• Deletetakesconstanttime,O(1).

• Just move one pointer!

» Note this assumes we are already at the element we want to delete. So we don‟t have to find that element which would add O(N).

• Inserttakesconstanttime,O(1).

• Just create a new element and move two pointers!

» Note this assumes we are already at the element we want to insert. So we don‟t have to find that element which would add O(N).

• Caninsert/deleteNelementsinO(N)time.

• Recall that the array implementation of the List ADT required O(N2). So this is much faster!

Dynamic array is good for direct, random access but the array has to be resized every time a new element is added. Faster than linked list as the allocation of memory dynamically for every node causes a lot of overhead especially if the list is small.

Linked list is good for sequential access and need not resize every time a new element is added due to dynamic allocation of memory.

arrays may be used to implement stacks and queues.

The Stack

To implement a stack we need to insert and remove from one end of the list. The end of the list may be chosen as to avoid any shifting in the list.

The Queue

To implement a queue we need to insert at one end and remove from the other end of the list.

For Link list insert or remove between the list

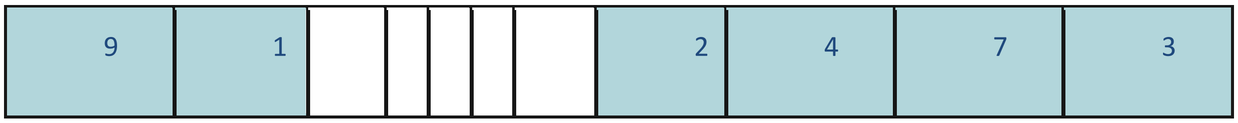
27- In a static array implementation of a queue, how do you know when the array is full? How do you know when the array is empty?

```
//The Queue is empty if isEmpty returns true

template<class T>

bool Queue <T>::isEmpty()const

{

bool  statuse;

if (numItems)

statuse =false;

else
statuse = true;

return  statuse;

}

//The Queue is fule if isFull returns true

template<class T>

bool Queue <T>::isFull()const

{

bool  statuse;

if (numItems<queueSize)

statuse =false;

else

statuse = true;

return  statuse;

}
```

In an array implementation of a queue, what is the problem of keeping the back index after the front index, and simply doubling the array when the back index reaches the last index?

There is just one problem. Nothing prevents the data values from wrapping around from the upper part of the array to the lower:

DataSize = 6 DataStart = 7 Data = flash to 9



To accommodate index values must be computed carefully. When an index wraps around the end of the array it must be altered to point to the start of the array. This is easily done by subtracting the capacity of the array. That is, suppose we try to index the fifth element in the picture above. We start by adding the index, 5, to the starting location, 7. The resulting sum is 12. But there are only eleven values in the collection. Subtracting 11 from 12 yields 1. This is the index for the value we seek.

28- what are difficulties in timing two algorithms to see which one is faster? How would you know which algorithm to use in your app? Here, I am looking for your thought process. Do you know the uses and limitation of Big O?

The Big O notation is a notation for the time complexity of an algorithm. It is a mathematical representation of the upper bound of the limit of the scaling factor of the algorithm. For example, if we double the size of an input array, by how much does the computing time increase?

The big O notation expresses the scaling of computing time and uses some sort of mixture between the upper bound and the limit of that scaling

29- What is the time complexity of inserting into the front of a linked list of size N, given the front of the list and a value to insert?

for inserting at the front is trivial: it is a single statement using no loops (and therefore is in the O(1) complexity class)

What is the time complexity of inserting into the back of a linked list of size N, given the front of the list and a value to insert?

the only node whose **next** is **null**. Then, the **next** of this node is changed to refer to a non-null value: a new node that becomes the rear of the linked list.
Notice that this code is NOT very efficient. It is O(N),

30- complete the definition of the following function:

```
//doubleArray(array,size)

//doubles the size of the passed dynamic array

//and copies all old elements into the new array

template <class dataType>

Voide doubleArray (dataType* &array, unsigned int &size)

{

newCap==2*size

if ( newCap == 0 )
```

```
	}

{
	Delete();
	return;
}

dataType * newData = new dataType[newCap];
bool * newInUse = new bool[newCap];
elementnumber= 0;

unsigned int upperLimit = capacity < newCap ? capacity : newCap;
for( unsigned int i = 0; i < upperLimit; ++i )
{
	newInUse[i] = in_use[i];
	if( in_use[i] )
	{
		elementnumber++;
		newData[i] = data[i];
	}
}
if( upperLimit < newCap )
{
	for( unsigned int i = upperLimit; i < newCap; ++i )
	{
		newInUse[i] = false;
	}
```

```
        }


        capacity = newCap;

        if ( data )

                delete [] data;

        if ( in_use )

                delete [] in_use;

        data = newData;

        in_use = newInUse;


}
```

31- complete the definition for the following function that inserts the passed value onto the top of the passed dynamic array. You may use the doubleArray function from the other question

```
//arrayStackInsert(array,size,value)

//insert the value

//on the 'top' of

// the dynamic array

template< class dataType>

void arrayStackInsert(dataType*&array,unsigned int &size,dataType const&value)

{


}
```

32- Write the definition to insert the passed value on top of the passed stack in the following function:

```
// Node<dataType>contains

// dataType data;

// Node <dataType> *next;

template<class dataType>

void listStackInsert(Node<dataType>*&stack, dataType const&value)

{

{

Node *newNode=nullptr;

Newnode = new  Node;

newNode->data = value;

if (is Empty())

{

Top = newNode;

newNode->next = nullptr;

}

Else

{

newNode->next = Top;

Top = newNode;

}


}
```