Answer Script

Question No. 1-a

Explain Stack and Heap memory.

Answer No. 1-a

The Stack memory allocation happens on contiguous blocks of memory. We call it a stack memory allocation because the allocation happens in the call stack. When a function or block of code is executed, the variables are pushed onto the stack, and when the function or block completes execution, the variables are popped off the stack. A stack is not flexible, the memory size allocated cannot be changed.

Heap memory is allocated in any random order and it is not managed automatically and its size can be changed during runtime. The variables stored in heap memory are accessed via pointers and can be shared between functions and blocks of code. Heap memory is flexible, and the allocated memory can be altered.

Question No. 1-b

Why do we need dynamic memory allocation? Explain with examples.

Answer No. 1-b

Dynamic memory allocation is required when we need to allocate memory at runtime instead of compile-time. This is useful when we don't know how much memory will be needed at the time of compilation, or when we want to allocate memory dynamically based on user input.
For example, if we want to create an array whose size is not known until runtime, we use dynamic memory allocation.
Another example, suppose you are writing a program to read a file with an unknown number of lines. Without dynamic memory allocation, you would have to select a fixed size for your array, which could be inefficient if the file contains many lines or wasteful if it contains only a few. With dynamic memory allocation, you can allocate memory as needed, ensuring that you don't waste memory while still having the ability to store the entire file.

Question No. 1-c
How to create a dynamic array? What are the benefits of it?

Answer No. 1-c

```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int size = 5;
    int *arr = new int[size];
    return 0;
}
```

One benefit of dynamic arrays is that they allow you to allocate memory based on runtime conditions, which makes the program more flexible and adaptable to different situations. For example, if you're writing a program to read data from a file, you might not know how many elements you'll need until you've read the whole file. A dynamic array allows you to allocate the memory you need based on the size of the data, rather than having to guess at a fixed size.
Another benefit of dynamic arrays is that they can be resized during runtime.

Finally, dynamic arrays can be more efficient in terms of memory usage, because you only allocate the memory you need. Static arrays, on the other hand, have a fixed size that is allocated at compile time, which can lead to wasted memory if the array is larger than necessary.

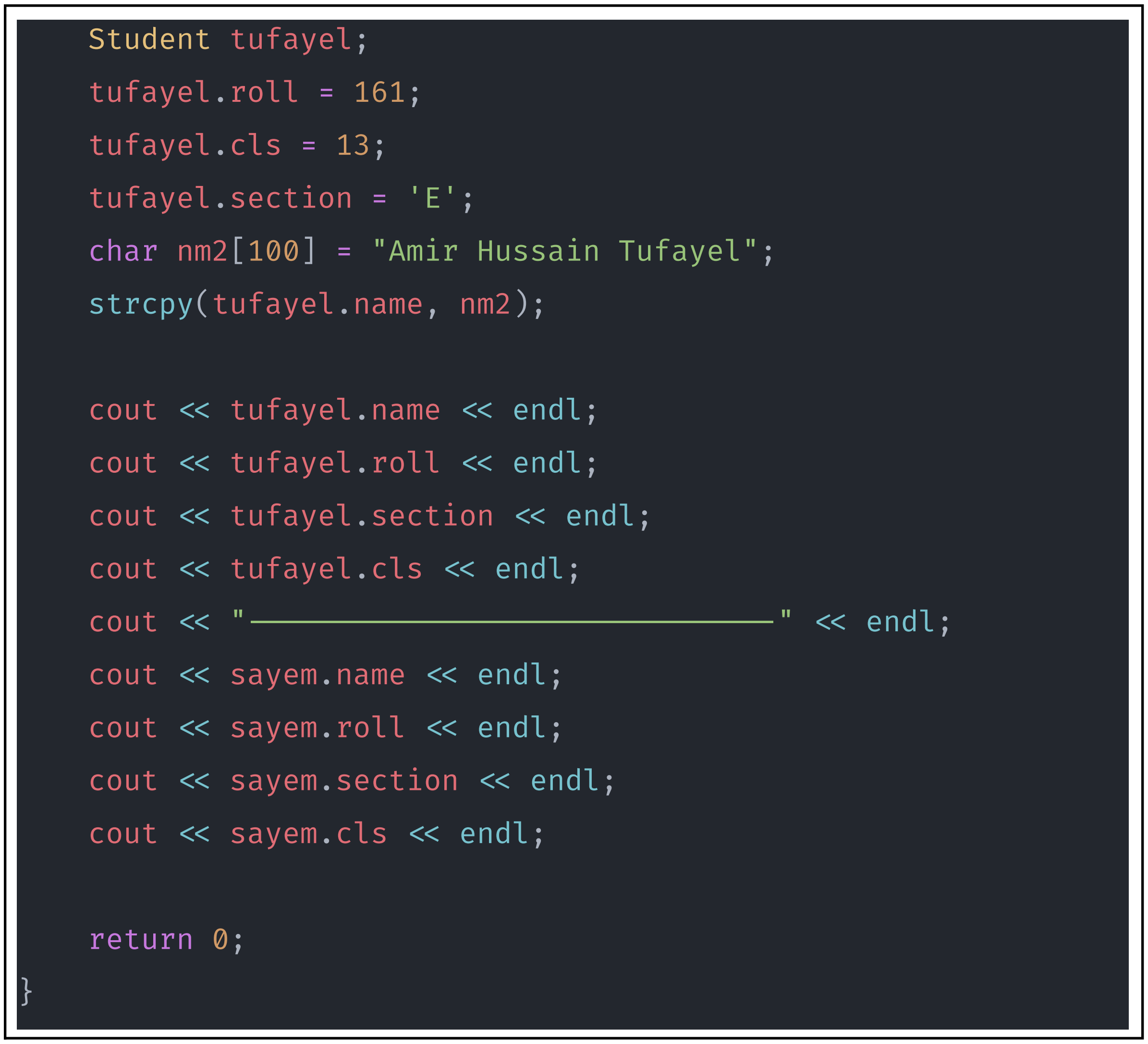
Question No. 2-a
How does class and object work? How to declare an object?
Answer No. 2-a

A class is a user-defined data type that encapsulates data and functions into a single entity that can be performed on objects.

An object is an instance of a class, created using the 'new' operator.

To declare an object, we first need to create a class and then use the class name to create an object. For example:

```cpp
#include <bits/stdc++.h>
using namespace std;
class Student//class
{
public:
    char name[100];
    int roll;
    int cls;
    char section;
};
int main()
{
    Student sayem;//object
    sayem.roll = 185;
    sayem.cls = 13;
    sayem.section = 'E';
    char nm1[100] = "Sheikh Eismile Ahmed Sayem";
    strcpy(sayem.name, nm1);
```

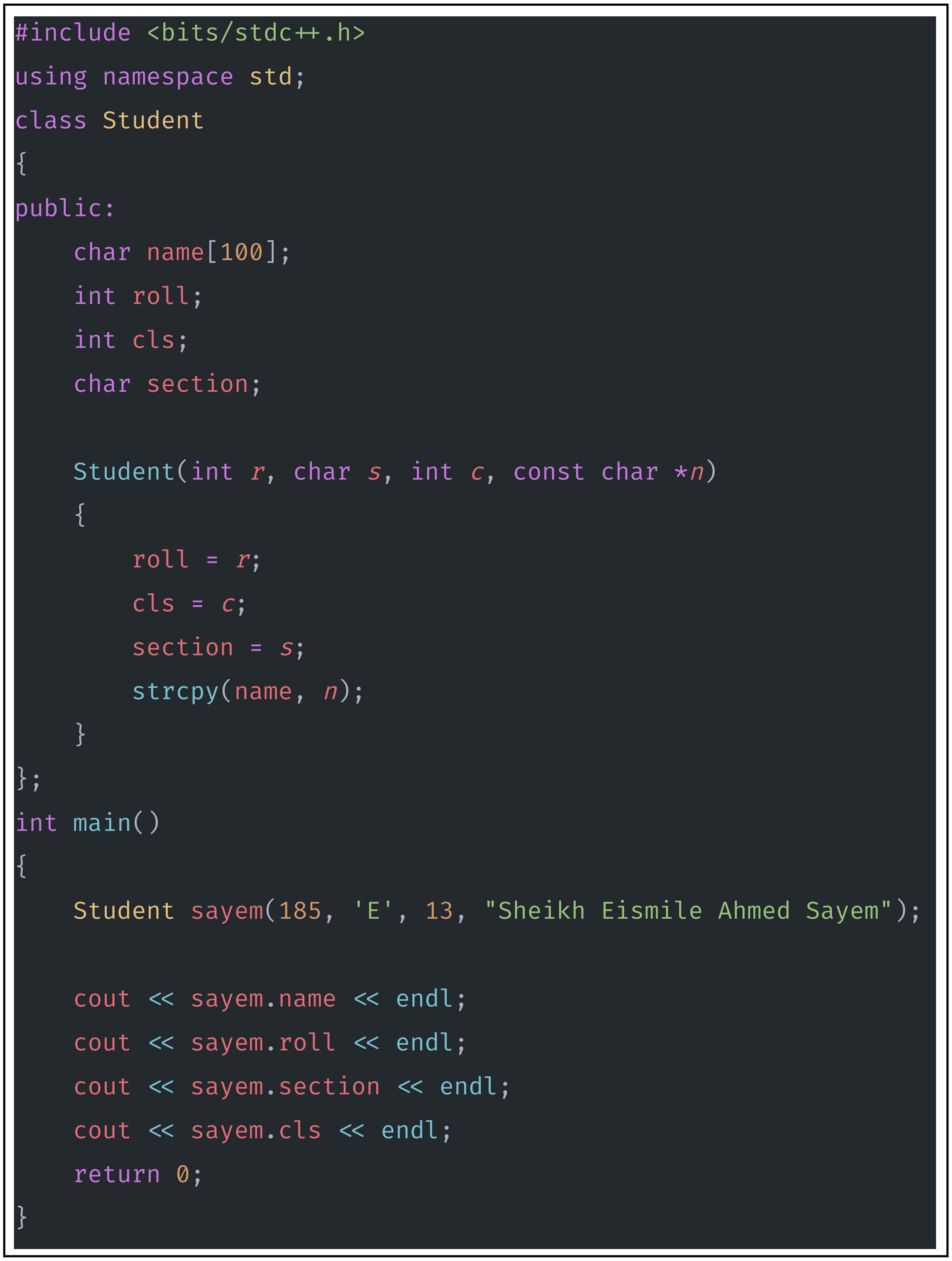


```cpp
    Student tufayel;
    tufayel.roll = 161;
    tufayel.cls = 13;
    tufayel.section = 'E';
    char nm2[100] = "Amir Hussain Tufayel";
    strcpy(tufayel.name, nm2);

    cout << tufayel.name << endl;
    cout << tufayel.roll << endl;
    cout << tufayel.section << endl;
    cout << tufayel.cls << endl;
    cout << "——————————————————————————————" << endl;
    cout << sayem.name << endl;
    cout << sayem.roll << endl;
    cout << sayem.section << endl;
    cout << sayem.cls << endl;

    return 0;
}
```

Question No. 2-b
What is a constructor and why do we need this? How to create a constructor show with an example.
Answer No. 2-b
A constructor is a special member function of a class that is called automatically when an object of that class is created. Its main purpose is to initialize the object's data members and perform any other necessary setup. We need constructors to ensure that objects are properly initialized and ready for use when they are created, and to handle memory allocation.



```cpp
#include <bits/stdc++.h>
using namespace std;
class Student
{
public:
    char name[100];
    int roll;
    int cls;
    char section;

    Student(int r, char s, int c, const char *n)
    {
        roll = r;
        cls = c;
        section = s;
        strcpy(name, n);
    }
};
int main()
{
    Student sayem(185, 'E', 13, "Sheikh Eismile Ahmed Sayem");

    cout << sayem.name << endl;
    cout << sayem.roll << endl;
    cout << sayem.section << endl;
    cout << sayem.cls << endl;
    return 0;
}
```

Question No. 2-c

Create a class named **Person** where the class will have properties name(string), height(float) and age(int). Make a constructor and create a dynamic object of that class and finally pass proper values using the constructor.

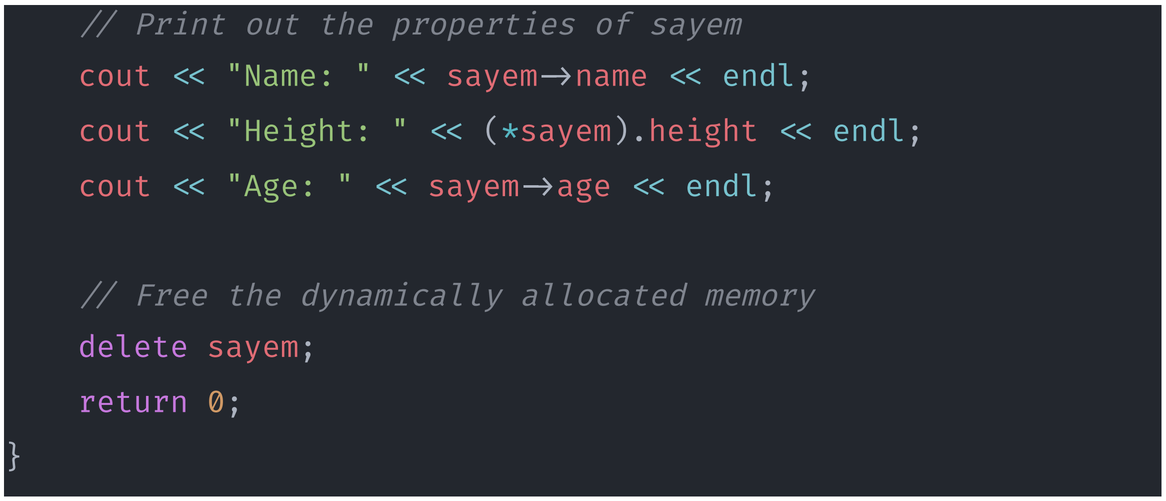
Answer No. 2-c

```cpp
#include <bits/stdc++.h>
using namespace std;

class Person
{
public:
    string name;
    float height;
    int age;

    Person(string n, float h, int a)
    {
        name = n;
        height = h;
        age = a;
    }
};

int main()
{
    // Create a dynamic object of the Person class and pass values using the constructor
    Person *sayem = new Person("Sheikh Eismile Ahmed Sayem", 1.63, 21);
```



```cpp
    // Print out the properties of sayem
    cout << "Name: " << sayem->name << endl;
    cout << "Height: " << (*sayem).height << endl;
    cout << "Age: " << sayem->age << endl;

    // Free the dynamically allocated memory
    delete sayem;
    return 0;
}
```

Question No. 3-a
What is the size that an object allocates to the memory?
Answer No. 3-a
The size of an object depends on the data members it contains. The total size of all the data inside the object is the memory size of the object. But if it's an empty object it will take 1 byte memory.

Question No. 3-b

Can you return a static object from a function? If yes, show with an example.

Answer No. 3-b

Yes.

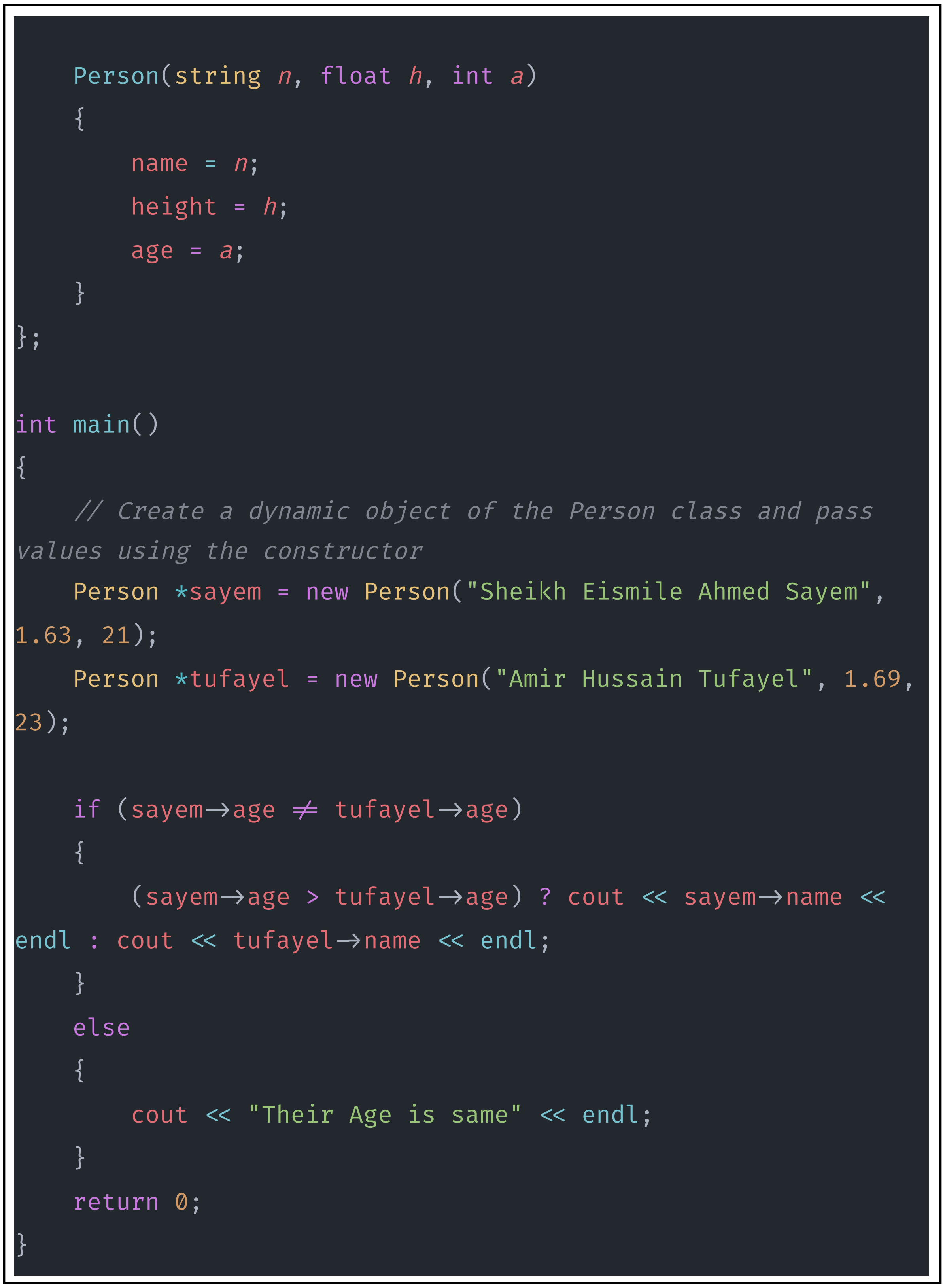
```cpp
#include <bits/stdc++.h>
using namespace std;
class Student
{
public:
    char name[100];
    int cls;
    int roll;
    char section;
    Student(int r, int c, char s, char *n)
    {
        roll = r;
        cls = c;
        section = s;
        strcpy(name, n);
    }
};
Student fun()
{
    char name[100] = "Sayem";
    Student s(185, 13, 'E', name);
    return s; // RVO-return value optimization
}
int main()
{
    Student sayem = fun();
    cout << sayem.name << endl;
    cout << sayem.roll << endl;
    cout << sayem.section << endl;
    cout << sayem.cls << endl;
    return 0;
}
```

Question No. 3-c
Why do we need -> (arrow sign)?
Answer No. 3-c
-> (arrow sign) is used to access the members of a dynamic object through a pointer.if we have a pointer to an object, we can access its data members using the arrow notation(->).We can also Access through dereferencing the pointer and using dot notation

Question No. 3-d
Create two objects of the **Person** class from question **2-c** and initialize them with proper value. Now compare whose age is greater, and print his/her name.
Answer No. 3-d

```
#include <bits/stdc++.h>
using namespace std;

class Person
{
public:
    string name;
    float height;
    int age;
```



```
    Person(string n, float h, int a)
    {
        name = n;
        height = h;
        age = a;
    }
};

int main()
{
    // Create a dynamic object of the Person class and pass
values using the constructor
    Person *sayem = new Person("Sheikh Eismile Ahmed Sayem",
1.63, 21);
    Person *tufayel = new Person("Amir Hussain Tufayel", 1.69,
23);

    if (sayem->age != tufayel->age)
    {
        (sayem->age > tufayel->age) ? cout << sayem->name <<
endl : cout << tufayel->name << endl;
    }
    else
    {
        cout << "Their Age is same" << endl;
    }
    return 0;
}
```