**Answer Script**

| Question No. 1-a |
| --- |
| Explain Stack and Heap memory. |
| Answer No. 1-a |
| **Stack Memory:**  Stack Memory is the memory where a variable stores its data at compile time of the program. Stack memory is also called static memory because once allocated the size of a variable can’t be changed. Some examples of variables stored in stack memory are - ***int a, char c*** etc.  **Heap Memory:**  On the other hand Heap memory is the memory where a variable stores its data in run time of the program. Heap memory is also called dynamic memory because we can change the size of the variable stored in heap memory any time we need. Some examples of variables stored in heap memory are - ***int \*a = new int, char \*c = new char*** etc. |

| Question No. 1-b |
| --- |
| Why do we need dynamic memory allocation? Explain with examples. |
| Answer No. 1-b |
| Dynamic memory allocation is the process of allocating memory at runtime. It is used when the size of the memory required is not known at compile time. In C++, dynamic memory allocation is done using the new and delete operators.   * Dynamic memory allocation allows us to allocate memory for objects that are too large to be stored on the stack. The stack is a limited area of memory that is used to store local variables. If an object is too large to be stored on the stack, it needs to be stored on the heap. * Dynamic memory allocation allows us to create objects that are of variable size. For example, if we want to create an array of integers, we can use the new keyword to create an array of any size.   #include <bits/stdc++.h>  using namespace std;  int main()  {  int \*arr = new int[6];  for (int i = 0; i < 6; i++)  {  cin >> arr[i];  }  for (int i = 0; i < 6; i++)  {  cout << arr[i] << "\t";  }  return 0;  }  In this above code we have created an dynamic array of size 6 which can be deleted later if we don’t need it using *delete[] arr;* |

| Question No. 1-c |
| --- |
| How to create a dynamic array? What are the benefits of it? |
| Answer No. 1-c |
| To create a dynamic array in C++, we can use the new keyword. The new keyword allocates memory on the heap and returns a pointer to the allocated memory. For example, the following code creates a dynamic array of 10 integers:  *int\* myArray = new int[10];*  Here *myArray* variable now points to a dynamic array of 10 integers.  Benefits of using dynamic arrays:   * Dynamic arrays are allocated on the heap, which is a larger area of memory. * Dynamic arrays can be resized at runtime.  We can add or remove elements from the array as needed. * Dynamic arrays are more efficient than static arrays. |

| Question No. 2-a |
| --- |
| How does class and object work? How to declare an object? |
| Answer No. 2-a |
| In C++, a class is a blueprint for creating objects. It defines the data and behavior of the objects that it creates. An object is an instance of a class. It has the data and behavior defined by the class.  To declare an object, we use the ***class name*** followed by a ***variable name***. For example, the following code declares an object of the Student class:  #include <bits/stdc++.h>  using namespace std;  class Student  {  public:  char name[100];  int roll;  char section;  };  int main()  {  ***Student*** *student*;  student.roll = 29;  student.section = 'A';  char nm[100] = "Maruf";  strcpy(student.name, nm);  return 0;  }  The ***student*** variable is now an object of the ***Student*** class. It has all the data and behavior defined by the ***Student*** class.  We can access the data and behavior of an object using the dot operator (.). For example, the following code prints the name of the student object:  *cout << student.name << endl;* |

| 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 |
| In C++, a constructor is a special member function that is called automatically when an object of its class is created. It is used to initialize the object's data members.   * Constructors are needed to initialize the data members of an object when it is created . * Constructors can be used to allocate memory dynamically for objects. If an object requires dynamic memory allocation, the constructor can handle the allocation and ensure that the object has the necessary resources to operate correctly.   *#include <bits/stdc++.h>*  *using namespace std;*  *class Student*  *{*  *public:*  *char name[100];*  *int roll;*  *char section;*  *//Constructor*  *Student(char \*n, int r, char s)*  *{*  *strcpy(name, n);*  *roll = r;*  *section = s;*  *}*  *};*  *int main()*  *{*  *Student maruf("Maruf Hasan", 69, 'A');*  *return 0;*  *}*  Here, the ***Student*** class defines a student object. The class has four data members: name, roll and section. The ***Student*** class also has a constructor, which is called when a **Student** object is created. The constructor takes four parameters: name, roll and section. The constructor uses these parameters to initialize the Student object's data members. |

| 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 |
| *#include <bits/stdc++.h>*  *using namespace std;*  *class Person*  *{*  *public:*  *char name[100];*  *float height;*  *int age;*  *Student(char \*n, float h, int a) //Constructor*  *{*  *strcpy(name, n);*  *height = h;*  *age = a;*  *}*  *};*  *int main()*  *{*  *Student maruf("Maruf Hasan", 6.00, 26); //Passing Values*  *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 allocated to memory in C++ is the sum of the sizes of its data members, plus any additional padding required for alignment.   * The size of a data member is determined by its type. For example, the size of an int is 4 bytes, and the size of a char is 1 byte.   *#include <bits/stdc++.h>*  *using namespace std;*  *class Student*  *{*  *public:*  *int roll;*  *char name[100];*  *};*  *int main()*  *{*  *Student student;*  *// Calculate the size of the student object.*  *int size = sizeof(student);*  *// Print the size of the student object.*  *cout << "The size of the student object is: " << size << endl;*  *return 0;*  *}*  In this example, the Student class has two data members: roll and name. The roll data member is of type int, which is 4 bytes in size. The name data member is of type char[100], which is 100 bytes in size. The size of the Student object is the sum of the sizes of its data members, plus any additional padding required for alignment. In this case, the size of the Student object is ***104 bytes***.  The size of an object can also be affected by the compiler's optimization settings. For example, if the compiler is set to optimize for speed, it may allocate less memory for an object than if it is set to optimize for size. |

| Question No. 3-b |
| --- |
| Can you return a static object from a function? If yes, show with an example. |
| Answer No. 3-b |
| Yes. We can return a static object from a function. Here is an example with code:  *#include <bits/stdc++.h>*  *using namespace std;*  *class Student*  *{*  *public:*  *char name[100];*  *int roll;*  *int age;*  *char section;*  *Student(char \*n, int r, int a, char s)*  *{*  *strcpy(name, n);*  *roll = r;*  *age = a;*  *section = s;*  *}*  *};*  *Student func()*  *{*  *char name[100] = "Maruf Hasan";*  *Student maruf(name, 2263, 25, 'A');*  *return maruf;*  *}*  *int main()*  *{*  *Student maruf = func();*  *cout << maruf.age << endl*  *<< maruf.roll << endl*  *<< maruf.section << endl*  *<< maruf.name << endl;*  *return 0;*  *}* |

| Question No. 3-c |
| --- |
| Why do we need -> (arrow sign)? |
| Answer No. 3-c |
| The arrow operator (->) is used in C++ to access the members of a class through a pointer. For example:  *#include <bits/stdc++.h>*  *using namespace std;*  *class Student*  *{*  *public:*  *int roll;*  *int age;*  *char section;*  *Student(int r, int a, char s)*  *{*  *roll = r;*  *age = a;*  *section = s;*  *}*  *};*  *int main()*  *{*  *Student \*maruf = new Student(2263, 25, 'A');*  *cout << maruf->age << endl;*  *cout << maruf->section << endl;*  *return 0;*  *}*  In this code, using the pointer we can access the ***age & section*** member from the dynamic object. |

| 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:*  *char name[100];*  *float height;*  *int age;*  *Person(char \*n, float h, int a) // Constructor*  *{*  *strcpy(name, n);*  *height = h;*  *age = a;*  *}*  *};*  *int main()*  *{*  *Person maruf("Maruf Hasan", 6.00, 26); // Passing Values*  *Person kona("Nishat Jahan Kona", 5.80, 25); // Passing Values*  *// Compare the ages of maruf and kona*  *if (maruf.age > kona.age)*  *{*  *cout << "Maruf is older than Kona" << endl;*  *}*  *else if (maruf.age < kona.age)*  *{*  *cout << "Kona is older than Maruf" << endl;*  *}*  *else*  *{*  *cout << "Maruf and Kona are the same age" << endl;*  *}*  *return 0;*  *}* |