|  |  |  |
| --- | --- | --- |
| **Question No** |  | **Question** |
| **1** | **a** | Explain Stack and Heap memory.  Answer: A memory that takes up the memory space it needs at the time of compilation of the code and does not take up any more space after it is run is called stack memory. On the other hand, heap memory is called dynamic memory, the memory that does not take up any specific space while compiling the code is called heap memory. |
| **b** | Why do we need dynamic memory allocation? Explain with examples.  Answer:  Dynamic memory allocation is an important concept in programming that allows us to dynamically allocate memory at runtime. This is especially necessary when we do not know the exact amount of memory required in advance or the memory requirements may change during program execution. In dynamic memory arrays can be incremented and arrays can be deleted.  Explain with examples:  #include<bits/stdc++.h>  using namespace std;  int main()  {      int n;      cin>>n;      int \*a=new int [n];      for(int i=0;i<n;i++)      {          cin>>a[i];      }      for(int i=0;i<n;i++)      {          cout<<a[i]<<" ";      }      delete[] a;      return 0;  }  In this example, the program prompts the user to enter the size of the array. Dynamic memory allocation is used to allocate memory for an integer array, based on input. Then the array is populated with values from 0 to size n. Finally, the allocated memory is released using delete[]. |
| **c** | How to create a dynamic array? What are the benefits of it?  Ans: A keyword called new is used to create a dynamic array.  #include<bits/stdc++.h>  using namespace std;  int main()  {      int n;      cin>>n;      int \*a=new int [n];      for(int i=0;i<n;i++)      {          cin>>a[i];      }      for(int i=0;i<n;i++)      {          cout<<a[i]<<" ";      }      return 0;  }  Advantages of dynamic arrays:  Dynamic arrays allow you to allocate memory for arrays of variable size at runtime. You can allocate memory for an array as needed. Dynamic arrays serve as the basis for implementing array transformations, stack data structures. Dynamic arrays allow you to create an array inside a function and pass it back to the calling function. The benefits of dynamic array that it can be extended array values and deleted array if not needed. |
| **2** | **a** | How does class and object work? How to declare an object?  Ans:  Class is a blue design. Which shows the structure and behavior of an object, like if we think of a car then that car has various behaviors like the number of wheels, how many cc of the engine, how many seats, etc. These are the components of the car blueprint. Based on this, various other types of objects are created.  Object declare:  Object Name Ashik  This is object line of code :      Person \*Ashik = new Person("Ashik",5.11,22);  This is full code.  #include<bits/stdc++.h>  using namespace std;  class Person  {      public:      char name[100];      float height;      int age;          Person(char \*n,float h,int a)          {              strcpy(name,n);              height=h;              age=a;          }  };  int main()  {        Person \*Ashik = new Person("Ashik",5.11,22);      cout<<(\*Ashik).name<<endl;      cout<<(\*Ashik).height<<endl;      cout<<(\*Ashik).age<<endl;      return 0;  } |
| **b** | What is a constructor and why do we need this? How to create a constructor show with an example.  Answer:  Constructor is a special method of c++. It is used in class functions. When we create an object, the constructor automatically calls the object. We use constructors when we need to repeatedly use the name, address, date, age, etc. on different objects.  Example:  #include<bits/stdc++.h>  #include<string.h>  using namespace std;  class Ticket  {      public:      char Passenger\_name[100];      int ticket\_number;      int ticket\_price;          Ticket(char\* pn,int tn,int tp)          {              strcpy(Passenger\_name,pn);              ticket\_number = tn;              ticket\_price = tp;          }  };  int main()  {      Ticket Ashik("Ashik Ahammed",01,250);      cout<<Ashik.Passenger\_name<<endl;      cout<<Ashik.ticket\_number<<endl;      cout<<Ashik.ticket\_price<<endl;      return 0;  } |
| **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:  #include<bits/stdc++.h>  using namespace std;  class Person  {      public:      char name[100];      float height;      int age;          Person(char \*n,float h,int a)          {              strcpy(name,n);              height=h;              age=a;          }  };  int main()  {        Person \*Ashik = new Person("Ashik",5.11,22);      cout<<(\*Ashik).name<<endl;      cout<<(\*Ashik).height<<endl;      cout<<(\*Ashik).age<<endl;      return 0;  } |
| **3** | **a** | What is the size that an object allocates to the memory?  Answer:  The size of an object in memory depends on the memory requirements of its member variables. An object occupies a certain amount of memory based on its member variable data type. In C++ sizeof operator is used to determine the size of an object. The sizeof operator returns the size of the object in bytes. |
| **b** | Can you return a static object from a function? If yes, show with an example.  Answer:  Yes, I return a static object from a function. This answer below.  #include<bits/stdc++.h>  using namespace std;  class Person  {      public:      char name[100];      float height;      int age;          Person(char \*n,float h,int a)          {              strcpy(name,n);              height=h;              age=a;          }  };  Person person1()  {      char name[100]="Ashik";      Person Ashik(name,5.11,23);      return Ashik;  }  int main()  {      Person Ashik=person1();      cout<<Ashik.name<<endl;      cout<<Ashik.height<<endl;      cout<<Ashik.age<<endl;      return 0;  } |
| **c** | Why do we need -> (arrow sign)?  Answer: In C++ arrow operator is used to access members variables or functions of an object that is pointed to by a pointer. When you have a pointer to an object, you need to use the arrow operator to access the members of that object through the pointer. This is necessary because the arrow operator combines both the dereference operator \* and the dot operator . |
| **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:  #include<bits/stdc++.h>  using namespace std;  class Person  {      public:      char name[100];      float height;      int age;          Person(char \*n,float h,int a)          {              strcpy(name,n);              height=h;              age=a;          }  };  int main()  {      Person Ashik("Ashik",5.11,23);      Person Ahammed("Ahammed",5.9,22);      if(Ashik.age > Ahammed.age)      cout<<(Ashik).name<<endl;      else if(Ashik.age < Ahammed.age)      cout<<(Ahammed).name<<endl;      else      cout<<"Same age"<<endl;      return 0;  } |