C++ Question & Answers

1. What is C++?

Ans: Released in 1985, C++ is an object-oriented programming language created by Bjarne Stroustrup. C++ maintains almost all aspects of the C language while simplifying memory management and adding several features - including a new data type known as a class (you will learn more about these later) - to allow object-oriented programming. C++ maintains the features of C which allowed for low-level memory access but also gives the programmer new tools to simplify memory management.

C++ used for:

C++ is a powerful general-purpose programming language. It can be used to create small programs or large applications. It can be used to make CGI scripts or console only DOS programs. C++ allows you to create programs to do almost anything you need to do. The creator of C++, Bjarne Stroustrup, has put together a partial list of applications written in C++.

2. How do you find out if a linked-list has an end? (i.e. the list is not a cycle)

Ans: You can find out by using 2 pointers. One of them goes to 2 nodes each time. The second one goes at 1 nodes each time. If there is a cycle, the one that goes 2 nodes each time will eventually meet the one that goes slower. If that is the case, then you will know the linked-list is a cycle.

3. What is the difference between realloc() and free()?

Ans: The free subroutine frees a block of memory previously allocated by the malloc subroutine. Undefined results occur if the Pointer parameter is not a valid pointer. If the Pointer parameter is a null value, no action will occur. The realloc subroutine changes the size of the block of memory pointed to by the Pointer parameter to the number of bytes specified by the Size parameter and returns a new pointer to the block. The pointer specified by the Pointer parameter must have been created with the malloc, calloc, or realloc subroutines and not been deallocated with the free or realloc subroutines. Undefined results occur if the Pointer parameter is not a valid pointer.

4. What is function overloading and operator overloading?

Ans: Function overloading: C++ enables several functions of the same name to be defined, as long as these functions have different sets of parameters (at least as far as their types are concerned). This capability is called function overloading. When an overloaded function is

called, the C++ compiler selects the proper function by examining the number, types, and order of the arguments in the call. Function overloading is commonly used to create several functions of the same name that perform similar tasks but on different data types. Operator overloading allows existing C++ operators to be redefined so that they work on objects of user-defined classes. Overloaded operators are syntactic sugar for equivalent function calls. They form a pleasant facade that doesn't add anything fundamental to the language (but they can improve understandability and reduce maintenance costs).

5. What is the difference between declaration and definition?

Ans: The declaration tells the compiler that at some later point we plan to present the definition of this declaration.

E.g.: void stars () //function declaration

The definition contains the actual implementation.

E.g.: void stars () // declarator
{
for(int j=10; j > =0; j--) //function body
cout << *;
cout <<>

6. What are the advantages of inheritance?

Ans: It permits code reusability. Reusability saves time in program development. It encourages the reuse of proven and debugged high-quality software, thus reducing problem after a system becomes functional.

7. How do you write a function that can reverse a linked-list?

Ans: void reverselist(void)
{
if(head==0)
 return;
if(head->next==0)
 return;
if(head->next==tail)
{
 head->next = 0;
 tail->next = head;
}
else

```
{
 node* pre = head;
 node* cur = head->next;
 node* curnext = cur->next;
 head->next = 0;
 cur-> next = head;
 for(; curnext!=0; )
 {
 cur->next = pre;
 pre = cur;
 cur = curnext;
 curnext = curnext->next;
Customer is king inc.,
Customer is king inc.,
 }
 curnext->next = cur;
}
}
```

8. What do you mean by inline function?

Ans: The idea behind inline functions is to insert the code of a called function at the point where the function is called. If done carefully, this can improve the application's performance in exchange for increased compile time and possibly (but not always) an increase in the size of the generated binary executables. Write a program that ask for user input from 5 to 9 then calculate the average

```
#include "iostream.h"
int main() {
int MAX = 4;
int total = 0;
int average;
int numb;
for (int i=0;i
{
cout << "Please enter your input between 5 and 9: ";
cin >> numb;
while ( numb<5>9) {
cout << "Invalid input, please re-enter: ";
cin >> numb;
}
total = total + numb;
}
```

```
average = total/MAX;
cout << "The average number is: " << average << "\n";
return 0;
}
```