HealthAsyst Placement Paper Questions

Q1. How do you find out if a linked-list has an end? (i.e. the list is not a cycle)

You can find out by using 2 pointers. One of them goes 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.

Q2. What is the difference between realloc() and free()?

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.

Q3. What is function overloading and operator overloading?

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 doesnt add anything fundamental to the language (but they can improve understandability and reduce maintenance costs).

Q4. What is the difference between declaration and definition?

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 <<>

What are the advantages of inheritance?

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.

Q5. How do you write a function that can reverse a linked-list?

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;
}
curnext->next = cur;
}
}
```

Q6. What do you mean by inline function?

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 applications performance in exchange for increased compile time and possibly (but not always) an increase in the size of the generated binary executables.

Q7. 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; icout << "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: " <<>return 0;
}
```

Q8. What is public, protected, private?

Public, protected and private are three access specifiers in C++.
Public data members and member functions are accessible outside the class.
Protected data members and member functions are only available to derived classes.
Private data members and member functions cant be accessed outside the class. However there is an exception can be using friend classes.
Write a function that swaps the values of two integers, using int* as the argument type.

```
void swap(int* a, int*b) {
int t;
t = *a;
*a = *b;
*b = t;
}
```

Q9. Tell how to check whether a linked list is circular.

Create two pointers, each set to the start of the list. Update each as follows:
while (pointer1)

```
{
pointer1 = pointer1->next;
pointer2 = pointer2->next; if (pointer2) pointer2=pointer2->next;
if (pointer1 == pointer2) {
print ("circular ");
}
```

```
}
```

OK, why does this work?

If a list is circular, at some point pointer2 will wrap around and be either at the item just before pointer1, or the item before that. Either way, its either 1 or 2 jumps until they meet.

Q10. What is polymorphism?

Polymorphism is the idea that a base class can be inherited by several classes. A base class pointer can point to its child class and a base class array can store different child class objects.

Q11. What is virtual constructors/destructors?

Answer1

Virtual destructors:

If an object (with a non-virtual destructor) is destroyed explicitly by applying the delete operator to a base-class pointer to the object, the base-class destructor function (matching the pointer type) is called on the object.
There is a simple solution to this problem declare a virtual base-class destructor.

This makes all derived-class destructors virtual even though they dont have the same name as the base-class destructor. Now, if the object in the hierarchy is destroyed explicitly by applying the delete operator to a base-class pointer to a derived-class object, the destructor for the appropriate class is called. Virtual constructor: Constructors cannot be virtual. Declaring a constructor as a virtual function is a syntax error.

Answer 2:

Virtual destructors: If an object (with a non-virtual destructor) is destroyed explicitly by applying the delete operator to a base-class pointer to the object, the base-class destructor function (matching the pointer type) is called on the object.

There is a simple solution to this problem – declare a virtual base-class destructor. This makes all derived-class destructors virtual even though they dont have the same name as the base-class destructor. Now, if the object in the hierarchy is destroyed explicitly by applying the delete operator to a base-class pointer to a derived-class object, the destructor for the appropriate class is called.

Virtual constructor:

Constructors cannot be virtual. Declaring a constructor as a virtual function is a syntax error.

Q12. Does c++ support multilevel and multiple inheritance?

Yes.

Q13. What are the advantages of inheritance?

• 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.

Q14. What is the difference between declaration and definition?

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<

Q15. What is the difference between an ARRAY and a LIST?

Answer1:

Array is collection of homogeneous elements.
List is collection of heterogeneous elements.

For Array memory allocated is static and continuous.
For List memory allocated is dynamic and Random.

Array: User need not have to keep in track of next memory allocation.
List: User has to keep in Track of next location where memory is allocated.

Answer2:

Array uses direct access of stored members, list uses sequencial access for members.

With Array you have direct access to memory position 5
Object x = a[5]; // x takes directly a reference to 5th element of array

With the list you have to cross all previous nodes in order to get the 5th node:
list mylist;
list::iterator it;

```
for( it = list.begin() ; it != list.end() ; it++ )
{
if( i==5)
{
x = *it;
break;
}
i++;
}
```

Q16. What is a template?

Templates allow to create generic functions that admit any data type as parameters and return value without having to overload the function with all the possible data types. Until certain point they fulfill the functionality of a macro. Its prototype is any of the two following ones:

template function_declaration; template function_declaration;

The only difference between both prototypes is the use of keyword class or typename, its use is indistinct since both expressions have exactly the same meaning and behave exactly the same way.

You have two pairs: new() and delete() and another pair : alloc() and free(). Explain differences between eg. new() and malloc()

Answer1
1.) “new and delete” are preprocessors while “malloc() and free()” are functions. [we dont use brackets will calling new or delete].
2.) no need of allocate the memory while using “new” but in “malloc()” we have to use “sizeof()”.
3.) “new” will initlize the new memory to 0 but “malloc()” gives random value in the new alloted memory location [better to use calloc()]

Answer2
new() allocates continous space for the object instace
malloc() allocates distributed space.
new() is castless, meaning that allocates memory for this specific type,
malloc(), calloc() allocate space for void * that is cated to the specific class type pointer.

Q17. What is the difference between class and structure?

Structure: Initially (in C) a structure was used to bundle different type of data types together to perform a particular functionality. But C++ extended the structure to contain functions also. The major difference is that all declarations inside a structure are by default public.
Class: Class is a successor of Structure. By default all the members inside the class are private.

Q18. What is RTTI?

Runtime type identification (RTTI) lets you find the dynamic type of an object when you have only a pointer or a reference to the base type. RTTI is the official way in standard C++ to discover the type of an object and to convert the type of a pointer or reference (that is, dynamic typing). The need came from practical experience with C++. RTTI replaces many homegrown versions with a solid, consistent approach.