1. What is the correct output of this code?

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main()
{
    vector<int> v = {1, 2, 3, 4, 5};
    v.erase(v.begin() + 2);
    v.insert(v.begin() + 1, 10);
    cout << v[2];
    return 0;
}
```

a) 2
b) 3
c) 10
d) 4

Explanation: প্রথমে erase(v.begin()+2) দ্বারা 3rd এলিমেন্ট (3) ডিলিট হয়, এখন ভেক্টর: {1,2,4,5} । তারপর insert(v.begin()+1, 10) দ্বারা 2nd পজিশনে 10 ইনসার্ট হয়। এখন ভেক্টর: {1,10,2,4,5}। v[2] হল 2।

2. What is the correct output of this code?

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main()
{
    vector<int> v(5, 1);
    for (int i = 0; i < 3; i++)
    {
        v.push_back(i * 2);
    }
    cout << v.size() << " " << v.capacity();
    return 0;
}
```

a) 8 10
b) 5 8
c) 8 8
d) 5 10

Explanation: প্রথমে ৫টি এলিমেন্ট (সব ১) তৈরি হয়। তারপর ৩টি এলিমেন্ট (0,2,4) পুশ করা হয়। মোট এলিমেন্ট ৮।

ভেক্টরের ক্যাপাসিটি ডাবলিং রুল অনুযায়ী ৫ থেকে ১০ হয়ে যায়।

3. What is the correct output of this code?

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
    vector<int> v = {2, 4, 6, 8};
    v.resize(6, 10);
    cout << v[4] << " " << v[5];
    return 0;
}
```

a) 0 0
b) 10 10
c) 8 10
d) Runtime error

Explanation: resize(6, 10) ভেক্টরের সাইজ 6 করে এবং নতুন এলিমেন্টগুলো 10 দিয়ে ফিল করে। মূল ভেক্টর ছিল {2,4,6,8}, নতুন ভেক্টর হবে {2,4,6,8,10,10}। তাই v[4] ও v[5] উভয়ই 10।

4. What is the correct output of this code?

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main()
{
    vector<int> v(5, 2);
    for (auto it = v.begin(); it != v.end(); it++)
    {
        cout << "Hello ";
        v.push_back(3);
    }
    return 0;
}
```

a) 5
b) Infinite Loop
c) 1
d) 0

Explanation: লুপটি ভেক্টরের মূল সাইজ 5, কিন্তু প্রতি লুপে push_back(3) ব্যবহার করে একটা করে আইটেম যুক্ত করছে। ফলে, ভেক্টর সাইজও বাড়তে থাকে তাই ইনফিনিটি লুপ তৈরী করবে।

5. What is the correct output of this code?

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main()
{
    vector<int> v = {1, 2, 3, 4, 5};
    auto it = v.begin() + 2;
    v.insert(v.begin() + 1, 10);
    cout << *it;
    return 0;
}
```

a) 2
b) 3
c) 10
d) Invalid

Explanation: ভেক্টরে insert/erase করার পর পূর্বের ইটারেটরগুলো ইনভ্যালিড হয়ে যায়। এটি ইনভ্যালিড আচরণ প্রদর্শন করতে পারে এবং গার্বেজ ভ্যালু প্রদর্শন করতে পারে।

6. What's the time complexity to find an element in a sorted singly linked list?

a) O(1)
b) O(log n)
c) O(n)
d) O(n log n)

Explanation: সবচেয়ে খারাপ ক্ষেত্রে, আমাদের সমস্ত N নোড check করতে হতে পারে।

7. What's the worst-case time complexity of inserting at the beginning of a vector with n elements?

a) O(1)
b) O(log n)
c) O(n)
d) $O(n^2)$

Explanation: সমস্ত index এ বিদ্যমান উপাদানগুলিকে ডানদিকে স্থানান্তরিত করতে হবে, যার জন্য O(n) সময় প্রয়োজন।

8. What is the time complexity to access the 5th element in a singly linked list with 10 nodes?

a) O(1)
b) O(log n)
c) O(n)
d) $O(n^2)$

Explanation: Linked lists এর কোনও নির্দিষ্ট অবস্থানে পৌঁছানোর জন্য হেড নোড থেকে ট্রাভার্সাল প্রয়োজন, যার ফলে linear time complexity দেখা দেয়।

9. For a singly linked list without a tail pointer, what is the time complexity of deleting the last node?

a) O(1)
b) O(n)
c) O(log n)
d) O(n²)

Explanation: টেইল পয়েন্টার ছাড়া, আপনাকে সম্পূর্ণ linked list অতিক্রম করে head থেকে শেষ নোডটির আগের নোড খুঁজে বের করতে হবে এবং এর পরবর্তী পয়েন্টার আপডেট করতে হবে।

10. What is the worst-case time complexity to search for a value in a sorted linked list?

a) O(1)
b) O(n)
c) O(log n)
d) O(n²)

Explanation: সবচেয়ে খারাপ ক্ষেত্রে, আপনাকে তালিকার প্রতিটি নোড check করতে হতে পারে।