Question 1 (1)

```
#include<bits/stdc++.h>
using namespace std ;

int main() {
        string input_word ;

        cout << "Enter  Your String :- " ;
        cin >> input_word ;

        vector<string> valid_words = {"apple" , "application" , "grape" , "pineapple" , "banana"} ;

        int max_correct = 0 ;
        int correct_idx = -1 ;
        int i = 0 ;

        while(i < valid_words.size()) {
                if(input_word.size()  != valid_words[i].size()) {
                        i++;
                        continue;
                }

                int current_correct = 0 ;

                for(int j = 0 ; j < input_word.size() ; j++) {
                        if(input_word[j] == valid_words[i][j]) {
                                current_correct++;
                        }

                }
                if(current_correct > max_correct ) {
                        max_correct = current_correct ;
                        correct_idx = i ;
                }

                i++;
        }

        if(correct_idx != -1) {
                cout << "Correct word is " << valid_words[correct_idx] << endl ;
        }

        else {
                cout << "No Closest Word Found" << endl;
        }

}
```

Question – 1 (2)

```
#include<bits/stdc++.h>
using namespace std ;

int main() {

        string order ;
        string s ;

        cout << "Enter the Order String :- " ;
        cin >> order ;
        cout << "Enter the Input String S :- " ;
        cin >> s ;

        unordered_map<char , int> priority_table ;

        int priority = order.size();
        for(char c : order) {
                priority_table[c] = priority ;
                priority--;
        }

        int i = 0 ;
        int j = 1 ;

        while(i < j && j < s.size()) {
                if(priority_table[j]  > priority_table[i]) {
                        char temp = s[i] ;
                        s[i] = s[j] ;
                        s[j] = temp ;
                        i++ ;
                        j = i+1;
                }

                j++;
        }

        cout << "Output is :- " << s << endl;

}
```

Question-2(1)

```
#include<bits/stdc++.h>
using namespace std ;

string decimal_to_binary(int num) {
	string ans = "" ;

	while(num != 0) {
		char curr = (num % 2 == 0 ? '0' : '1') ;
		ans += curr ;

		num /= 2 ;
	}

	reverse(ans.begin() , ans.end()) ;

	return ans;
}

int main() {
	int a;
	int b ;

	cout << "Enter Number - 1 :- " ;
	cin >> a ;

	cout << "Enter Number - 2 :- " ;
	cin >> b ;

	string n1 = decimal_to_binary(a) ;
```

```
        string n2 = decimal_to_binary(b) ;

        for(int i = 1 ; i <= 8 - n1.size(); i++) {
                n1 = "0" + n1 ;
        }

        for(int i = 1 ; i <= 8 - n2.size(); i++) {
                n2 = "0" + n2 ;
        }

        int flip = 0 ;
        int no_of_bits = n1.size();

        for(int i = 0 ; i < no_of_bits ; i++) {
                if(n1[i] != n2[i]) {
                        flip++;
                }
        }

        cout << "Flips Required are :- " << flip << endl;
}
```

Question 3(B)

```
#include <bits/stdc++.h>
using namespace std;

int findCutVertex (vector<vector<int>> &graph, int node) {
        int min_degree = INT_MAX ;
        int cut_vertices = -1 ;
        for(int i = 0 ; i < node ; i++) {
                int curr_degree = 0 ;

                for(int j = 0 ; j < node ; j++) {
                        if(graph[i][j] != 9999 && graph[i][j] != 0) {
                                curr_degree++;
                        }
                }

                if(curr_degree < min_degree) {
                        cut_vertices = i ;
                }
        }

        return cut_vertices;
}


int main()
{
  int node;

  cout << "Enter The Number of Nodes :- ";
  cin >> node;

  vector<vector<int>> graph(node, vector<int>(node, 9999));

  cout << "Enter The Adjacency Matrix :- " << endl;

  for (int i = 0; i < node; i++)
  {
        for (int j = 0; j < node; j++)
        {
            cout << "Enter The Edge from " << i << " to " << j << " (9999 for no edge) :- ";
            cin >> graph[i][j];
        }
    }

    int cut_vertices = findCutVertex(graph, node);

    if(cut_vertices != -1) {
            cout << "The Cut Vertex Is " << cut_vertices << endl ;
            }

            else {
                        cout << "All Vertex Having the Strong Connectiveity." << endl ;
            }

    return 0;
}
```

Question 4 :-

```
#include<bits/stdc++.h>
using namespace std ;

bool isPalindrome(string &s , int start , int end) {
            string curr(s.begin() + start , s.end() + (end + 1)) ;

            int size = end - start + 1;

            for(int i = 0 ; i <= size / 2 ; i++) {
```

```
        if(s[i] != s[size - i - 1]) {
            return false ;
        }
    }

    return true;
}

int Total_Palindrome_Subsequence(string &s) {
    int total = 0 ;

    for(int i = 0 ; i < s.size() ; i++) {
        for(int j = i + 1 ; j < s.size()  ; j++) {
            bool ans = isPalindrome(s, i , j) ;
            total += ans ;
        }
    }

    return total ;
}

int main() {
    string s ;

    cout << "Enter the String :- " ;
    cin >> s ;

    int total_Palindrome = Total_Palindrome_Subsequence(s) ;

    cout <<"The Total Palindrome Subsequences are " << total_Palindrome << endl;
```

```
        return 0;
}
```