2. (b) Test 2

Test Summary

- No. of Sections: 2
- No. of Questions: 3
- Total Duration: 35 min

Section 1 - Coding Proficiency

Section Summary

- No. of Questions: 2
- Duration: 20 min

Additional Instructions:
None

Q1. **Merge Sort Using Pointers**
Write a program to merge sort using pointers

Input Format

Input contains the size of the array and the values

Output Format

Print the sorted values

Constraints

Array size may vary
Apply merge sort to sort the values

Sample Input

```
5
78 64 23 6 93
```

Sample Output

```
6 23 64 78 93
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q2. **Remove Vowels**
Given a string str, write a program to eliminate all the vowels from it.

The list of vowels In the English alphabet is : {a,e,i,o,u,A,E,I,0.U}

The Input to the function eliminateVowelString shall consist of a string str (containing only English letters) and returns a pointer to a string which does not contain vowels.

Example:
Input =”abcdefghijklmnopqrstuvwxyz” l
0utput="bcdfghjklmnpqrstvwxyz”

Useful Commands:

Strlen() is used to calculate the length of the string. The statement -int len = strlen(str);
Returns the length of the string str

Input Format

Input contains the string

Output Format

print the altered string

Constraints

1<= string_length<=1000

Sample Input

Sample Output

```
gAztkTJkCcmUVphMtGEDcWMMLSccLPvrMyLKTYYhkCYfZAiTDJKuSE
```

```
gztkTJkCcmVphMtGDcWMMLSccLPvrMyLKTYYhkCYfZTDJKSfSwnntW
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Section 2 - Essay Writing

Section Summary

- No. of Questions: 1
- Duration: 15 min

Additional Instructions:
None

Q1. **ESSAY WRITING**

Write a response explaining your preference. Justify your opinion with suitable examples.

Directions
Is Life better In a small town or a big city?

Keywords

Answer Key & Solution

Section 1 - Coding Proficiency

Q1

Test Case

Input

```
511
511 768 477 384 994 199 146 694 62 308 200 821 2
```

Output

```
3 4 6 6 9 19 19 20 21 22 23 24 25 26 27 27 29 
```

Weightage - 10

Input

```
791
857 930 822 595 514 46 764 878 48 323 883 669 91
```

Output

```
0 1 2 4 4 5 8 9 9 9 9 10 11 11 12 12 14 14 17
```

Weightage - 10

Input

```
148
873 128 699 958 344 3 971 290 808 320 487 840 25
```

Output

```
3 12 17 22 29 35 56 57 60 69 104 107 116 125 12
```

Weightage - 10

Input

```
688
111 624 192 345 475 754 582 863 672 236 891 753
```

Output

```
0 1 6 9 9 15 17 17 20 21 21 21 24 25 27 29 32 
```

Weightage - 10

Input

```
760
798 575 152 469 881 503 542 886 169 205 723 317
```

Output

```
0 0 1 4 4 5 6 6 10 10 10 12 13 14 17 18 22 23
```

Weightage - 10

Input

```
563
641 721 154 672 847 273 659 273 1 302 800 460 88
```

Output

```
1 3 4 13 15 19 21 22 24 25 27 29 30 31 33 33 35
```

Weightage - 10

Input

```
159
695 417 355 398 577 74 233 723 302 100 88 860 25
```

Output

```
1 18 44 45 55 56 61 63 71 74 84 88 91 100 102 1
```

Weightage - 5

Weightage - 5

Input

```
521
382 456 505 285 660 447 156 691 46 897 115 435 1
```

Output

```
2 3 3 4 7 9 9 10 12 14 18 19 21 22 22 23 27 29
```

Weightage - 10

Input

```
882
449 299 616 45 977 291 567 995 894 99 243 370 98
```

Output

```
2 3 4 9 9 11 12 13 13 15 15 15 16 16 17 18 20
```

Weightage - 10

Input

```
293
91 144 943 204 877 848 502 991 651 445 877 958 5
```

Output

```
3 5 8 11 12 13 21 24 25 25 31 33 37 38 39 39 40
```

Weightage - 10

Input

```
10
96 38 87 84 36 58 12 56 29 4
```

Output

```
4 12 29 36 38 56 58 84 87 96
```

Weightage - 5

Sample Input

```
5
78 64 23 6 93
```

Sample Output

```
6 23 64 78 93
```

Solution

```c
/* C program for Merge Sort */
#include<stdlib.h>
#include<stdio.h>
#include<malloc.h>

// Merges two subarrays of arr[].
// First subarray is arr[l..m]
// Second subarray is arr[m+1..r]
void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 =  r - m;

    /* create temp arrays */
    int L[n1], R[n2];


    /* Copy data to temp arrays L[] and R[] */
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1+ j];

    /* Merge the temp arrays back into arr[l..r]*/
    i = 0; // Initial index of first subarray
    j = 0; // Initial index of second subarray
    k = l; // Initial index of merged subarray
    while (i < n1 && j < n2)
    {
        if (L[i] <= R[j])
        {
            arr[k] = L[i];
            i++;
        }
        else
        {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    /* Copy the remaining elements of L[], if there
       are any */
    while (i < n1)
    {
        arr[k] = L[i];
        i++;
        k++;
    }

    /* Copy the remaining elements of R[], if there
       are any */
    while (j < n2)
    {
        arr[k] = R[j];
        j++;
        k++;
    }
}

/* l is for left index and r is right index of the
   sub-array of arr to be sorted */
void mergeSort(int arr[], int l, int r)
{
    if (l < r)
    {
        // Same as (l+r)/2, but avoids overflow for
        // large l and h
        int m = l+(r-l)/2;

        // Sort first and second halves
        mergeSort(arr, l, m);
        mergeSort(arr, m+1, r);

        merge(arr, l, m, r);
    }
}

/* UTILITY FUNCTIONS */
/* Function to print an array */
void printArray(int A[], int size)

{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", A[i]);
   //printf("\n");
}

/* Driver program to test above functions */
int main()
{
    int *arr = NULL ;
    //[] = {12, 11, 13, 5, 6, 7};
    int arr_size ,ctr;
    //= sizeof(arr)/sizeof(arr[0]);

    //printf("Given array is \n");
    //printArray(arr, arr_size);
     scanf("%d",&arr_size);
     arr = (int *)malloc(sizeof(int) * arr_size);
     for( ctr =0 ; ctr < arr_size ; ctr++)
        scanf("%d" , &arr[ctr]);
    mergeSort(arr, 0, arr_size - 1);

    //printf("\nSorted array is \n");
    printArray(arr, arr_size);
    return 0;
}



/* C program for Merge Sort */
#include<stdlib.h>
#include<stdio.h>
#include<malloc.h>

// Merges two subarrays of arr[].
// First subarray is arr[l..m]
// Second subarray is arr[m+1..r]
void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 =  r - m;

    /* create temp arrays */
    int L[n1], R[n2];

    /* Copy data to temp arrays L[] and R[] */
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1+ j];

    /* Merge the temp arrays back into arr[l..r]*/
    i = 0; // Initial index of first subarray
    j = 0; // Initial index of second subarray
    k = l; // Initial index of merged subarray
    while (i < n1 && j < n2)
    {
        if (L[i] <= R[j])
        {
            arr[k] = L[i];
            i++;
        }
        else
        {

            arr[k] = R[j];
            j++;
        }
        k++;
    }

    /* Copy the remaining elements of L[], if there
       are any */
    while (i < n1)
    {
        arr[k] = L[i];
        i++;
        k++;
    }

    /* Copy the remaining elements of R[], if there
       are any */
    while (j < n2)
    {
        arr[k] = R[j];
        j++;
        k++;
    }
}

/* l is for left index and r is right index of the
   sub-array of arr to be sorted */
void mergeSort(int arr[], int l, int r)
{
    if (l < r)
    {
        // Same as (l+r)/2, but avoids overflow for
        // large l and h
        int m = l+(r-l)/2;

        // Sort first and second halves
        mergeSort(arr, l, m);
        mergeSort(arr, m+1, r);

        merge(arr, l, m, r);
    }
}

/* UTILITY FUNCTIONS */
/* Function to print an array */
void printArray(int A[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", A[i]);
   //printf("\n");
}

/* Driver program to test above functions */
int main()
{
    int *arr = NULL ;
    //[] = {12, 11, 13, 5, 6, 7};
    int arr_size ,ctr;
    //= sizeof(arr)/sizeof(arr[0]);

    //printf("Given array is \n");
    //printArray(arr, arr_size);
     scanf("%d",&arr_size);
     arr = (int *)malloc(sizeof(int) * arr_size);

     for( ctr =0 ; ctr < arr_size ; ctr++)
```

```
            scanf("%d" , &arr[ctr]);
        mergeSort(arr, 0, arr_size - 1);

        //printf("\nSorted array is \n");
        printArray(arr, arr_size);
        return 0;
    }
```

Q2

Test Case

Input

```
JUEyYAaXtrDKhuBaYWwNiFtcxKxhfHviPhGhXYGKhSekKckzpY
```

Output

```
JyYXtrDKhBYWwNFtcxKxhfHvPhGhXYGKhSkKckzpY
```

Weightage - 5

Input

```
gNHSeCDuJHRtVuBLvggSgLqLDgCZXZTjVFzBTVQBqphWtaShwU
```

Output

```
gNHSCDJHRtVBLvggSgLqLDgCZXZTjVFzBTVQBqphWtShw
```

Weightage - 5

Input

```
mTdxSkXRxqUziCXqNzUBPntZGtfRJDvJKryQAzycbEQFtJqWyuSi
```

Output

```
mTdxSkXRxqzCXqNzBPntZGtfRJDvJKryQzycbQFtJqWySHZpHhZK
```

Weightage - 10

Input

```
fRNVwzHGCBAjwECLzqYEwBUUwPVKJbaNPAaPMJmgLwCTrHaLVTGS
```

Output

```
fRNVwzHGCBjwCLzqYwBwPVKJbNPPMJmgLwCTrHLVTGSGVCBJGTBh
```

Weightage - 10

Input

```
ADmwJzCEAwLFqUVHmDxcnVxXwKvWHbcinDTYMzTBcwMEqcjzvNxw
```

Output

```
DmwJzCwLFqVHmDxcnVxXwKvWHbcnDTYMzTBcwMqcjzvNxwrRhbGF
```

Weightage - 10

Input

```
jcUqLLYdjRtGtRPBegywpiqJSkJWxxZinNdtKSigGyawkKkYQWaK
```

Output

```
jcqLLYdjRtGtRPBgywpqJSkJWxxZnNdtKSgGywkKkYQWKQkRmNGd
```

Weightage - 10

Input

Output

```
SGHVWJjPwUUBUWJqzwQMJcFtCZrvgxfRbFmjvBChgLDawxtQVKdK
```

```
SGHVWJjPwBWJqzwQMJcFtCZrvgxfRbFmjvBChgLDwxtQVKdKTxhT
```

Weightage - 10

Input

Output

```
czYmKYgupUnbDugczyBFgjJQbfxRjkSbwyUjHJuLwvgCiKxpnmuC
```

```
czYmKYgpnbDgczyBFgjJQbfxRjkSbwyjHJLwvgCKxpnmCZXywKvY
```

Weightage - 10

Input

Output

```
uiEbvsEOeOoAEiEuUOEbEoaIttOeUaUEoUAbaEOsIAAUvEEoOoub
```

```
bvsbttbsvbbvvbstbbtvbstsbsssstvvssvsbbvvtvbsvtsvbttvv
```

Weightage - 10

Input

Output

```
IsaaEssiOtIsAvAAEEOoiUteaiOuisbEEIAbOUOvvavbaoIisvsu
```

```
ssstsvtsbbvvvbsvsbbvtvbtvbbbtbvsvsbbvbtstsbstsbvbtss
```

Weightage - 10

Input

Output

```
ebuvuOoeAuIttooaouabtuEuOEsostbveiOAOAvEIiIIbtviOiOE
```

```
bvttbtsstbvvbtvssvbvvvttssststvtbststvttssbsbsstvvtt
```

Weightage - 10

Sample Input

Sample Output

```
gAztkTJkCcmUVphMtGEDcWMMLSccLPvrMyLKTYYhkCYfZAiTDJKu
```

```
gztkTJkCcmVphMtGDcWMMLSccLPvrMyLKTYYhkCYfZTDJKSfSwnn
```

Solution

Header

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
#include <malloc.h>


char* mystrchr(char* str, char ch)
{
```

Header

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
#include <malloc.h>


char* mystrchr(char* str, char ch)
{
    int index;
    for(index = 0 ; str[index] ; index++)
    {
        if(str[index] == ch)
            return str+index;
    }
    return NULL;

}


char * eliminateVowelString(char *str)
{
    int index = 0, update = 0;
   char*  ptr = NULL,vowels[] = "AEIOUaeiou";
   for(index = 0 ; str[index] ; index++)
   {
       ptr = mystrchr(vowels, str[index]);
        if(ptr == NULL)
            str[update++] = str[index];
   }
   str[update] = '\0';
   //printf("%s",str);
   return str;
}
```

```
    int index;
    for(index = 0 ; str[index] ; index++)
    {
        if(str[index] == ch)
            return str+index;
    }
    return NULL;

}


char * eliminateVowelString(char *str)
{
    int index = 0, update = 0;
   char*  ptr = NULL,vowels[] = "AEIOUaeiou";
   for(index = 0 ; str[index] ; index++)
   {
       ptr = mystrchr(vowels, str[index]);
        if(ptr == NULL)
            str[update++] = str[index];
   }
   str[update] = '\0';
   //printf("%s",str);
   return str;
}
```

Footer

```
int main()
{
    int test,len,ctr;
    char  str[1000];

        scanf("%s",str);
   eliminateVowelString(str);
   printf("%s",str);
       return 0;
}
```

Footer

```
int main()
{
    int test,len,ctr;
    char  str[1000];

        scanf("%s",str);
   eliminateVowelString(str);
   printf("%s",str);
       return 0;
}
```

Section 2 - Essay Writing

Q1 Sample Essay

No Essay

Keywords

LIFE , BETTER, SMALL , TOWN, BIG, CITY,