**Q1.**

#*include*<stdio.h>

#*include*<stdlib.h>

struct node

{

    int data;

    struct node \*next;

};

struct node \**create*(struct node \*head)

{

    struct node \*temp, \*newnode;

    int n, i;

*printf*("Enter the number of nodes: ");

*scanf*("%d", &n);

*for* (i = 0; i < n; i++)

    {

        newnode = (struct node \*)*malloc*(sizeof(struct node));

*printf*("Enter the data: ");

*scanf*("%d", &newnode->data);

        newnode->next = *NULL*;

*if* (head == *NULL*)

        {

            head = newnode;

            temp = newnode;

        }

*else*

        {

            temp->next = newnode;

            temp = newnode;

        }

    }

*return* head;

}

struct node \**merge*(struct node \*head1, struct node \*head2)

{

    struct node \*temp1, \*temp2;

    temp1 = head1;

    temp2 = head2;

*while* (temp1 != *NULL* && temp2 != *NULL*)

    {

*if* (temp1->data > temp2->data)

        {

            struct node \*newnode = (struct node \*)*malloc*(sizeof(struct node));

            newnode->data = temp2->data;

            newnode->next = temp1;

            head1 = newnode;

            temp1 = head1;

            temp2 = temp2->next;

        }

*else* *if* (temp1->data < temp2->data)

        {

*if* (temp1->next != *NULL*)

            {

*if* (temp1->next->data > temp2->data)

                {

                    struct node \*newnode = (struct node \*)*malloc*(sizeof(struct node));

                    newnode->data = temp2->data;

                    newnode->next = temp1->next;

                    temp1->next = newnode;

                    temp2 = temp2->next;

                }

*else*

                {

                    temp1 = temp1->next;

                }

            }

*else*

            {

                struct node \*newnode = (struct node \*)*malloc*(sizeof(struct node));

                newnode->data = temp2->data;

                newnode->next = *NULL*;

                temp1->next = newnode;

                temp2 = temp2->next;

            }

        }

    }

*return* head1;

}

void *display*(struct node \*head)

{

    struct node \*temp;

    temp = head;

*while* (temp != *NULL*)

    {

*printf*("%d ", temp->data);

        temp = temp->next;

    }

*printf*("");

}

int *main*()

{

    struct node \*head1 = *NULL*, \*head2 = *NULL*;

    head1 = *create*(head1);

    head2 = *create*(head2);

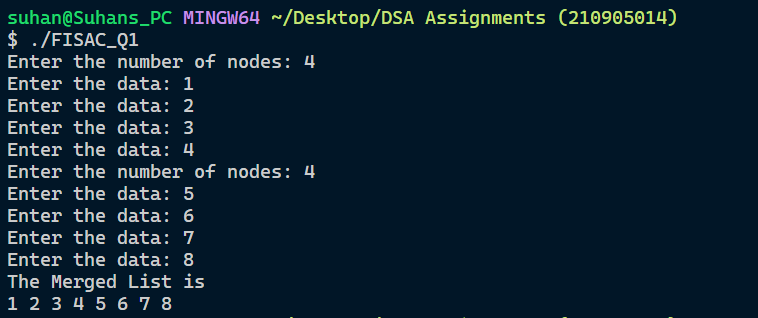
    head1 = *merge*(head1, head2);

*printf*("The Merged List is \n");

*display*(head1);

*return* 0;

}

**OUTPUT**

**Q2.**

#*include* <stdio.h>

#*include* <stdlib.h>

struct node

{

    int data;

    struct node \*next;

};

struct node \**create*(struct node \*head)

{

    struct node \*temp, \*newnode;

    int n, i;

*printf*("Enter the number of nodes: ");

*scanf*("%d", &n);

*for* (i = 0; i < n; i++)

    {

        newnode = (struct node \*)*malloc*(sizeof(struct node));

*printf*("Enter the data: ");

*scanf*("%d", &newnode->data);

        newnode->next = *NULL*;

*if* (head == *NULL*)

        {

            head = newnode;

            temp = newnode;

        }

*else*

        {

            temp->next = newnode;

            temp = newnode;

        }

    }

*return* head;

}

void *display*(struct node \*head)

{

    struct node \*temp;

    temp = head;

*while* (temp != *NULL*)

    {

*printf*("%d ", temp->data);

        temp = temp->next;

    }

*printf*("");

}

struct node\* *copy*(struct node\* head)

{

    struct node\* current = head;// *used to iterate over the original list*

    struct node\* newList = *NULL*;// *head of the new list*

    struct node\* tail = *NULL*;// *point to the last node in a new list*

*while* (current != *NULL*)

    {

        // *special case for the first new node*

*if* (newList == *NULL*)

        {

            newList = (struct node\*)*malloc*(sizeof(struct node));

            newList->data = current->data;

            newList->next = *NULL*;

            tail = newList;

        }

*else* {

            tail->next = (struct node\*)*malloc*(sizeof(struct node));

            tail = tail->next;

            tail->data = current->data;

            tail->next = *NULL*;

        }

        current = current->next;

    }

*return* newList;

}

// *Function to rotate the list to the right by k nodes*

void *rotateRight*(struct node\*\* head, int k)

{

    // *base case*

*if* (\*head == *NULL* || k == 0) {

*return*;

    }

    // *find the length of the list*

    int n = 1;

    struct node\* current = \*head;

*while* (current->next != *NULL*)

    {

        current = current->next;

        n++;

    }

    // *connect the last node to the head node*

    current->next = \*head;

    // *find the (n - k % n)-th node and make it the new head*

    struct node\* prev = *NULL*;

    current = \*head;

*for* (int i = 0; i < n - k % n; i++)

    {

        prev = current;

        current = current->next;

    }

    // *set the new head*

    \*head = current;

    // *set the next pointer of the last node to NULL*

    prev->next = *NULL*;

}

// *Function to rotate the list to the left by k nodes*

void *rotateLeft*(struct node\*\* head, int k)

{

    // *base case*

*if* (\*head == *NULL* || k == 0) {

*return*;

    }

    // *find the length of the list*

    int n = 1;

    struct node\* current = \*head;

*while* (current->next != *NULL*)

    {

        current = current->next;

        n++;

    }

    // *connect the last node to the head node*

    current->next = \*head;

    // *find the (n - k % n)-th node and make it the new head*

    struct node\* prev = *NULL*;

    current = \*head;

*for* (int i = 0; i < k % n; i++)

    {

        prev = current;

        current = current->next;

    }

    // *set the new head*

    \*head = current;

    // *set the next pointer of the last node to NULL*

    prev->next = *NULL*;

}

int *main*()

{

    struct node \*head = *NULL*;

    struct node \*head1 = *NULL*;

    int k;

    head = *create*(head);

*printf*("Enter the value of k: ");

*scanf*("%d", &k);

*printf*("LIST: ");

*display*(head);

    head1 = *copy*(head);

*rotateRight*(&head, k);

*printf*("\nRight rotate: ");

*display*(head);

*rotateLeft*(&head1, k);

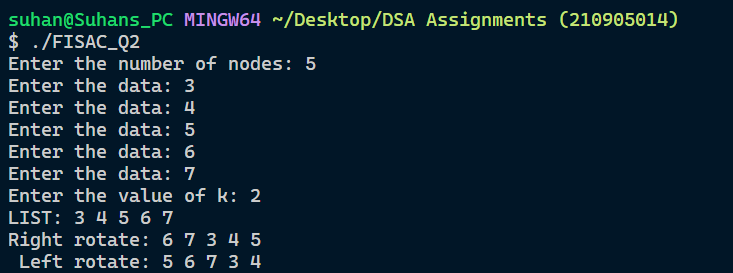
*printf*("\nLeft rotate: ");

*display*(head1);

*return* 0;

}

**OUTPUT**

****

**Q3.**

#*include* <stdio.h>

#*include* <stdlib.h>

struct node

{

    int data;

    struct node \*next;

};

struct node \**create*(struct node \*head)

{

    struct node \*temp, \*newnode;

    int n, i;

*printf*("Enter the number of nodes: ");

*scanf*("%d", &n);

*for* (i = 0; i < n; i++)

    {

        newnode = (struct node \*)*malloc*(sizeof(struct node));

*printf*("Enter the data: ");

*scanf*("%d", &newnode->data);

        newnode->next = *NULL*;

*if* (head == *NULL*)

        {

            head = newnode;

            temp = newnode;

        }

*else*

        {

            temp->next = newnode;

            temp = newnode;

        }

    }

*return* head;

}

void *display*(struct node \*head)

{

    struct node \*temp;

    temp = head;

*while* (temp != *NULL*)

    {

*printf*("%d ", temp->data);

        temp = temp->next;

    }

*printf*("");

}

struct node \**fibonacci*(struct node \*head)

{

    struct node \*temp, \*newnode;

    int n, i, a = 0, b = 1, c;

*printf*("Enter n: ");

*scanf*("%d", &n);

*for* (i = 0; i < n; i++)

    {

        newnode = (struct node \*)*malloc*(sizeof(struct node));

        newnode->data = a;

        newnode->next = *NULL*;

*if* (head == *NULL*)

        {

            head = newnode;

            temp = newnode;

        }

*else*

        {

            temp->next = newnode;

            temp = newnode;

        }

        c = a + b;

        a = b;

        b = c;

    }

*return* head;

}

int *main*()

{

    struct node \*head = *NULL*;

    head = *fibonacci*(head);

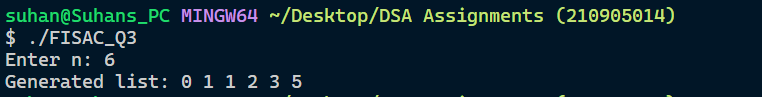
*printf*("Generated list: ");

*display*(head);

*return* 0;

}

**OUTPUT**

****

**Q4.**

#*include*<stdio.h>

#*include*<stdlib.h>

#*include*<string.h>

struct Node{

    char \*name;

    int roll\_no;

    int marks;

    struct Node \*next;

};

void *display*(struct Node \*first)

{

    struct Node \*temp = first;

*printf*("Contents of list are\n");

*while*(temp != *NULL*)

    {

*printf*("%s,%d,%d\n",temp->name,temp->roll\_no,temp->marks);

    temp = temp->next;

    }

}

struct Node \**insert\_rear*(char \*n,int roll,int mark,struct Node \*first)

{

   struct Node \*ptr;

   ptr = (struct Node \*)*malloc*(sizeof(struct Node \*));

   ptr->name =  (char \*)*malloc*(10\*sizeof(char));

   ptr->name = n;

   ptr->roll\_no = roll;

   ptr->marks = mark;

*if*(first == *NULL*)

   {

    ptr->next = *NULL*;

*return* ptr;

   }

   struct Node \*temp = first;

*while*(temp->next != *NULL*)

   {

    temp = temp->next;

   }

   temp->next = ptr;

   ptr->next = *NULL*;

*return* first;

}

void *swap*(struct Node \*first,struct Node \*second)

{

    int temproll;

    char \*t;

    t = (char \*)*malloc*(sizeof(char \*));

    int tempmark;

    t = first->name;

    temproll = first->roll\_no;

    tempmark = first->marks;

    first->name = second->name;

    first->roll\_no = second->roll\_no;

    first->marks = second->marks;

    second->name = t;

    second->roll\_no = temproll;

    second->marks = tempmark;

}

struct Node \**sorthighest*(struct Node \*first,int n)

{

    int temp;

    struct Node \*ptr = first;

*for*(int i=0;i<n-1;i++)

    {

*while*(ptr->next != *NULL*)

   {

*if*(ptr->marks<ptr->next->marks)

          {

*swap*(ptr,ptr->next);

            ptr = ptr->next;

          }

*else*{

            ptr = ptr->next;

          }

   }

   ptr = first;

    }

*return* first;

 }

int *main*()

{

    struct Node \*head;

    head = (struct Node \*)*malloc*(sizeof(struct Node \*));

    head = *NULL*;

*printf*("Enter number of students\n");

    int n,i;

*scanf*("%d",&n);

   // *char s[n][15];*

    char \*s1;

    int roll,mark;

*for*(i=0;i<n;i++)

        {

*printf*("Enter name of student %d\n",(i+1));

            s1 =  (char \*)*malloc*(10\*sizeof(char));

*scanf*("%s",s1);

*printf*("Enter Roll number of student %d\n",(i+1));

*scanf*("%d",&roll);

*printf*("Enter marks of student %d\n",(i+1));

*scanf*("%d",&mark);

            head = *insert\_rear*(s1,roll,mark,head);

        }

*display*(head);

*printf*("After sorting according to marks :\n");

    head = *sorthighest*(head,n);

    //*printf("After sorting according to marks :\n");*

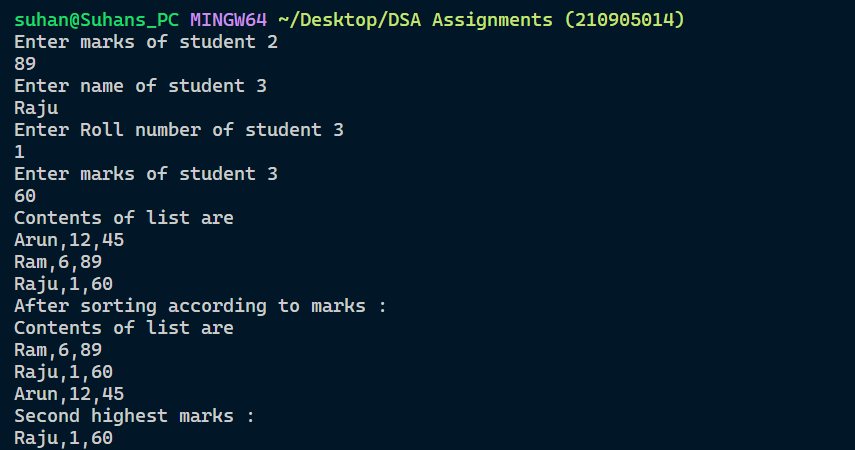
*display*(head);

*printf*("Second highest marks :\n%s,%d,%d",head->next->name,head->next->roll\_no,head->next->marks);

*return* 0;

}

**OUTPUT**

****

**Q5.**

#*include* <stdio.h>

#*include* <stdlib.h>

struct Node

{

  int data;

  struct Node \*next;

};

void *display*(struct Node \*first)

{

*if* (first == *NULL*)

  {

*printf*("List is empty\n");

*exit*(0);

  }

*printf*("Contents of list are\n");

*while* (first != *NULL*)

  {

*printf*("%d\n", first->data);

    first = first->next;

  }

}

struct Node \**insert\_rear*(int ele, struct Node \*first)

{

  struct Node \*ptr;

  ptr = (struct Node \*)*malloc*(sizeof(struct Node \*));

  ptr->data = ele;

*if* (first == *NULL*)

  {

    ptr->next = *NULL*;

*return* ptr;

  }

  struct Node \*temp = first;

  ;

*while* (temp->next != *NULL*)

  {

    temp = temp->next;

  }

  temp->next = ptr;

  ptr->next = *NULL*;

*return* first;

}

struct Node \**delete\_front*(struct Node \*first)

{

*if* (first == *NULL*)

  {

*printf*("List is empty\n");

*exit*(0);

  }

  struct Node \*t = first->next;

*free*(first);

*return* t;

}

struct Node \**delete\_rear*(struct Node \*first)

{

  struct Node \*temp1 = first;

  struct Node \*temp2 = first->next;

*while* (temp2->next != *NULL*)

  {

    temp2 = temp2->next;

    temp1 = temp1->next;

  }

  temp1->next = *NULL*;

*free*(temp2);

*return* first;

}

struct Node \**deletemiddle*(struct Node \*head, struct Node \*ptr)

{

  struct Node \*temp = head->next;

  struct Node \*temp1 = head;

*while* (temp != ptr)

  {

    temp = temp->next;

    temp1 = temp1->next;

  }

  temp1->next = temp->next;

*free*(temp);

*return* head;

}

struct Node \**delete\_occurence*(struct Node \*head, int k)

{

  int key;

  key = k \* k;

  struct Node \*ptr = head->next;

  struct Node \*ptr1 = head;

  struct Node \*ptr2 = head;

*while* (ptr != *NULL*)

  {

*if* (ptr->data == key && ptr->next == *NULL*)

    {

      head = *delete\_rear*(head);

      ptr = ptr->next;

      ptr1 = ptr1->next;

    }

*else* *if* (ptr->data == key && ptr->next != *NULL*)

    {

      head = *deletemiddle*(head, ptr);

      ptr = ptr->next;

      ptr1 = ptr1->next;

    }

*else*

    {

      ptr = ptr->next;

      ptr1 = ptr1->next;

    }

  }

*if* (ptr2->data == key)

  {

    head = *delete\_front*(head);

  }

*return* head;

}

int *main*()

{

  struct Node \*head;

  head = (struct Node \*)*malloc*(sizeof(struct Node \*));

  int choice, n, t, k;

*do*

  {

*printf*("Enter 1 to create linked list\n");

*printf*("Enter 2 delete occurences of linked list\n");

*printf*("Enter 3 to display linked list\n");

*printf*("Enter 4 to terminate\n");

*scanf*("%d", &choice);

*switch* (choice)

    {

*case* 1:

      head = *NULL*;

*printf*("Enter number of elements \n");

*scanf*("%d", &n);

*printf*("Enter elements \n");

*for* (int i = 0; i < n; i++)

      {

*scanf*("%d", &t);

        head = *insert\_rear*(t, head);

      }

*break*;

*case* 2:

*printf*("Enter key\n");

*scanf*("%d", &k);

      head = *delete\_occurence*(head, k);

*break*;

*case* 3:

*display*(head);

*break*;

*case* 4:

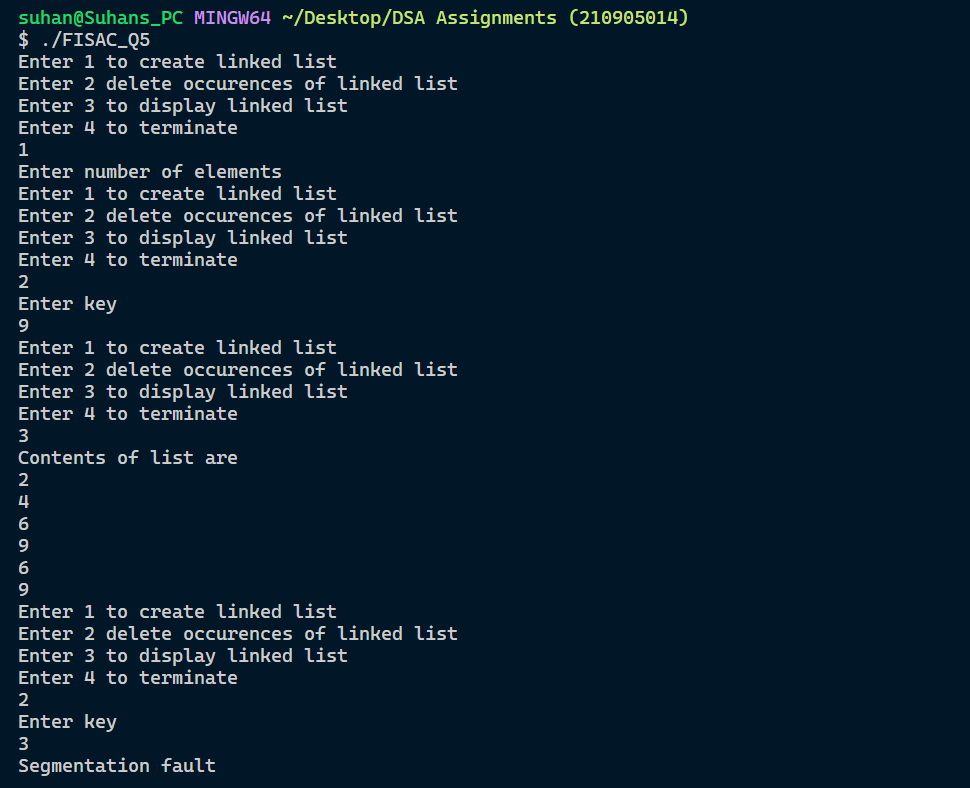
*exit*(0);

    }

  } *while* (1);

}

**OUTPUT**

****

**Q6.**

#*include*<stdio.h>

#*include*<stdlib.h>

struct Node{

    int data;

    struct Node \*next;

};

void *display*(struct Node \*first)

{

*if*(first == *NULL*)

    {

*printf*("List is empty\n");

*exit*(0);

    }

*printf*("Contents of list are\n");

*while*(first != *NULL*)

    {

*printf*("%d\n",first->data);

     first = first->next;

    }

}

struct Node \**insert\_rear*(int ele,struct Node \*first)

{

   struct Node \*ptr;

   ptr = (struct Node \*)*malloc*(sizeof(struct Node \*));

   ptr->data = ele;

*if*(first == *NULL*)

   {

    ptr->next = *NULL*;

*return* ptr;

   }

   struct Node \*temp = first;;

*while*(temp->next != *NULL*)

   {

    temp = temp->next;

   }

   temp->next = ptr;

   ptr->next = *NULL*;

*return* first;

}

struct Node \**cubelist*(struct Node \*head,struct Node \*head1,struct Node \*head2,int n)

{

    int temp;

   struct Node \*ptr = head;

*while*(ptr != *NULL*){

*if*(ptr->data % 2 == 0)

   {

     temp = ptr->data\*ptr->data\*ptr->data;

     head2 = *insert\_rear*(temp,head2);

     ptr = ptr->next;

   }

*else*{

    temp = ptr->data\*ptr->data\*ptr->data;

     head1 = *insert\_rear*(temp,head1);

     ptr = ptr->next;

   }

   }

*if*(n==0)

     {

*return* head1;

     }

*else*

     {

*return* head2;

     }

   }

int *main*()

{

    struct Node \*head;

    struct Node \*head1;

    struct Node \*head2;

    head = (struct Node \*)*malloc*(sizeof(struct Node \*));

    head1 = (struct Node \*)*malloc*(sizeof(struct Node \*));

    head2 = (struct Node \*)*malloc*(sizeof(struct Node \*));

*printf*("Enter number of elements \n");

    head = *NULL*;

    head1=*NULL*;

    head2=*NULL*;

    int n1,n2,t;

*scanf*("%d",&n1);

*printf*("Enter elements \n");

*for*(int i=0;i<n1;i++)

    {

*scanf*("%d",&t);

        head = *insert\_rear*(t,head);

    }

*display*(head);

*printf*("Odd list after cubing: \n");

    head1 = *cubelist*(head,head1,head2,0);

*display*(head1);

    head2 = *cubelist*(head,head1,head2,1);

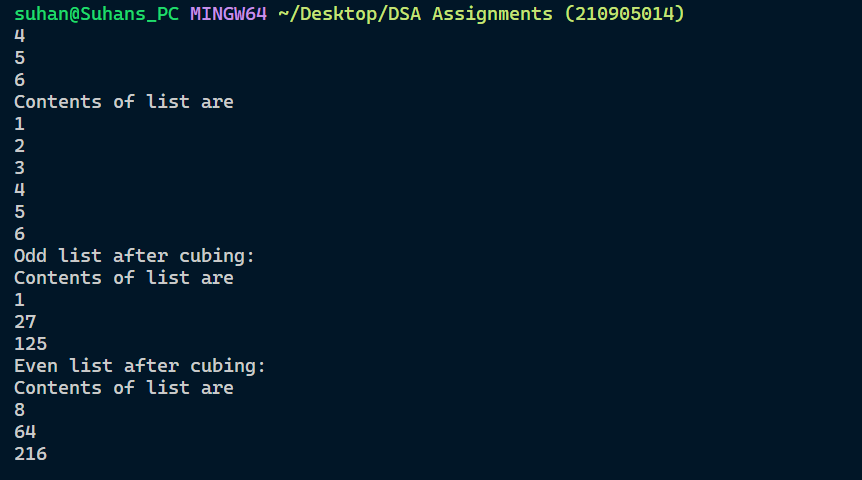
*printf*("Even list after cubing: \n");

*display*(head2);

*return* 0;

}

**OUTPUT**

****

**Q7.**

#*include*<stdio.h>

#*include*<stdlib.h>

#*include*<string.h>

struct Node{

    char \*data;

    struct Node \*next;

};

void *display*(struct Node \*first)

{

    struct Node \*temp = first;

*while*(temp != *NULL*)

    {

*printf*("%s\n",temp->data);

     temp = temp->next;

    }

}

struct Node \**insert\_rear*(char \*ele,struct Node \*first)

{

   struct Node \*ptr;

   ptr = (struct Node \*)*malloc*(sizeof(struct Node \*));

   ptr->data =  (char \*)*malloc*(10\*sizeof(char));

   ptr->data = ele;

*if*(first == *NULL*)

   {

    ptr->next = *NULL*;

*return* ptr;

   }

   struct Node \*temp = first;

*while*(temp->next != *NULL*)

   {

    temp = temp->next;

   }

   temp->next = ptr;

   ptr->next = *NULL*;

*return* first;

}

int *isPalindrome*(char \*str, int start, int end){

*if*(*NULL* == str || start < 0 || end < 0)

    {

*return* 0;

    }

*if*(start >= end)

         {

*return* 1;

         }

*if*(str[start] == str[end]){

*return* *isPalindrome*(str, start + 1, end - 1);

     }

*return* 0;

 }

 struct Node \**palindrome*(struct Node \*head,struct Node \*head1)

 {

    struct Node \*temp1 = head;

    int n;

      char \*s1;

*while*(temp1 != *NULL*)

    {

*if*(*isPalindrome*(temp1->data,0,*strlen*(temp1->data)-1))

       {

         s1 =  (char \*)*malloc*(10\*sizeof(char));

         s1 = temp1->data;

        head1 = *insert\_rear*(s1,head1);

         temp1 = temp1->next;

       }

*else*

       {

        temp1 = temp1->next;

       }

    }

*return* head1;

 }

 struct Node \**notpalindrome*(struct Node \*head,struct Node \*head2)

 {

    struct Node \*temp1 = head;

    int n;

      char \*s1;

*while*(temp1 != *NULL*)

    {

*if*(!(*isPalindrome*(temp1->data,0,*strlen*(temp1->data)-1)))

       {

         s1 =  (char \*)*malloc*(10\*sizeof(char));

         s1 = temp1->data;

        head2 = *insert\_rear*(s1,head2);

        temp1 = temp1->next;

       }

*else*

       {

        temp1 = temp1->next;

       }

    }

*return* head2;

 }

int *main*()

{

    struct Node \*head;

    struct Node \*head1;

    struct Node \*head2;

    head = (struct Node \*)*malloc*(sizeof(struct Node \*));

    head1 = (struct Node \*)*malloc*(sizeof(struct Node \*));

    head2 = (struct Node \*)*malloc*(sizeof(struct Node \*));

    head = *NULL*;

    head1 = *NULL*;

    head2 = *NULL*;

*printf*("Enter number of words\n");

    int n,i;

*scanf*("%d",&n);

    char \*s1;

*printf*("Enter words\n");

*for*(i=0;i<n;i++)

        {

            s1 =  (char \*)*malloc*(10\*sizeof(char));

*scanf*("%s",s1);

            head = *insert\_rear*(s1,head);

        }

*printf*("List of words :\n");

*display*(head);

    head1=*palindrome*(head,head1);

*printf*("List containing palindromes :\n");

*display*(head1);

     head2=*notpalindrome*(head,head2);

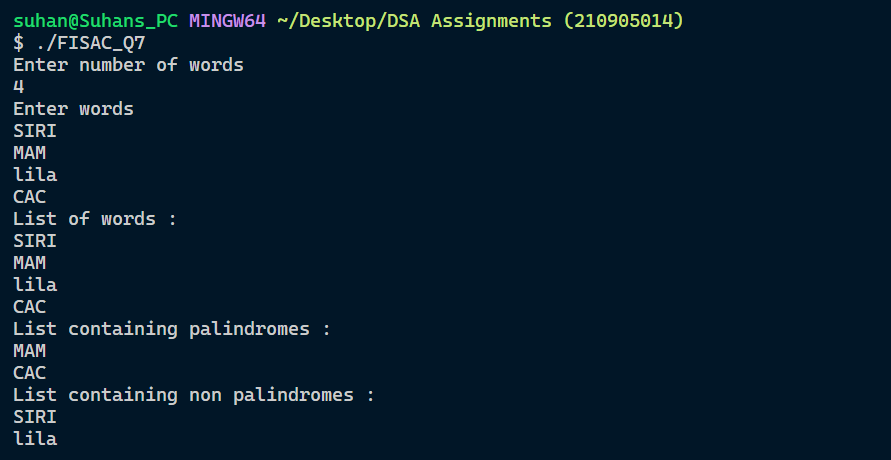
*printf*("List containing non palindromes :\n");

*display*(head2);

*return* 0;

}

**OUTPUT**

****

**Q8.**

#*include*<stdio.h>

#*include*<stdlib.h>

struct Node{

    int data;

    struct Node \*next;

};

void *display*(struct Node \*first)

{

*if*(first == *NULL*)

    {

*printf*("List is empty\n");

*exit*(0);

    }

*printf*("Contents of list are\n");

*while*(first != *NULL*)

    {

*printf*("%d\n",first->data);

     first = first->next;

    }

}

struct Node \**insert\_rear*(int ele,struct Node \*first)

{

   struct Node \*ptr;

   ptr = (struct Node \*)*malloc*(sizeof(struct Node \*));

   ptr->data = ele;

*if*(first == *NULL*)

   {

    ptr->next = *NULL*;

*return* ptr;

   }

   struct Node \*temp = first;;

*while*(temp->next != *NULL*)

   {

    temp = temp->next;

   }

   temp->next = ptr;

   ptr->next = *NULL*;

*return* first;

}

struct Node \**sortlist*(struct Node \*head,struct Node \*head1,struct Node \*head2,int n)

{

   struct Node \*ptr = head;

   int i = 1;

*while*(ptr != *NULL*)

   {

*if*(i%2 != 0)

    {

        head1 = *insert\_rear*(ptr->data,head1);

        i++;

        ptr=ptr->next;

    }

*else*{

         head2 = *insert\_rear*(ptr->data,head2);

        i++;

        ptr=ptr->next;

    }

   }

*if*(n==0)

     {

*return* head1;

     }

*else*

     {

*return* head2;

     }

   }

  // *return head1;*

int *main*()

{

    struct Node \*head;

    struct Node \*head1;

    struct Node \*head2;

    head = (struct Node \*)*malloc*(sizeof(struct Node \*));

    head1 = (struct Node \*)*malloc*(sizeof(struct Node \*));

    head2 = (struct Node \*)*malloc*(sizeof(struct Node \*));

*printf*("Enter number of elements \n");

    head = *NULL*;

    head1=*NULL*;

    head2=*NULL*;

    int n1,n2,t;

*scanf*("%d",&n1);

*printf*("Enter elements \n");

*for*(int i=0;i<n1;i++)

    {

*scanf*("%d",&t);

        head = *insert\_rear*(t,head);

    }

*display*(head);

*printf*("Odd list : \n");

    head1 = *sortlist*(head,head1,head2,0);

*display*(head1);

    head2 = *sortlist*(head,head1,head2,1);

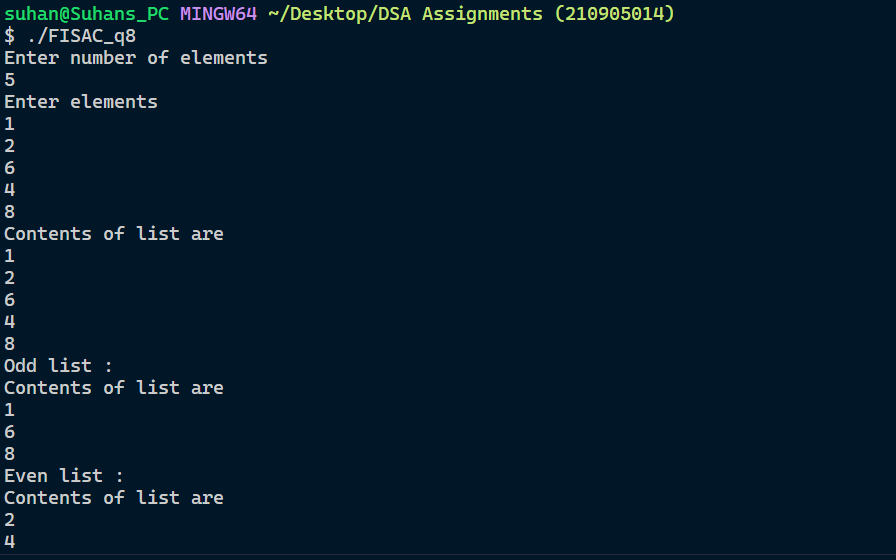
*printf*("Even list : \n");

*display*(head2);

*return* 0;

}

**OUTPUT**

****

**Q9.**

#*include* <stdio.h>

#*include* <stdlib.h>

#*include* <string.h>

// *Create a struct Node*

struct Node

{

    char data;

    struct Node \*next;

};

// *pointer to the node*

typedef struct Node \*Nodeptr;

// *Create Singly linked List from the given name*

Nodeptr *create*(char name*[]*)

{

    // *create empty head node*

    Nodeptr head = *NULL*;

    // *Iterate through all the characters in the given name*

*for* (int index = 0; index < *strlen*(name); index++)

    {

        // *create a new node*

        Nodeptr newNode = *malloc*(sizeof(struct Node));

        newNode->data = name[index];

        newNode->next = *NULL*;

        // *if head is empty, make the newNode as head*

*if* (head == *NULL*)

        {

            head = newNode;

        }

*else*

        {

            // *Iterate through the list and find last node in the list*

            Nodeptr temp = head;

*while* (temp->next != *NULL*)

            {

                temp = temp->next;

            }

            // *insert the new node at the end*

            temp->next = newNode;

        }

    }

*return* head;

}

void *allVowels*(Nodeptr \*head)

{

*if* (head != *NULL*)

    {

        // *Remove vowles from head node*

        char currentChar = (\*head)->data;

*while* ((\*head) != *NULL* && (currentChar == 'a' || currentChar == 'A' || currentChar == 'e' || currentChar == 'E' || currentChar == 'i' || currentChar == 'I' || currentChar == 'o' || currentChar == 'O' || currentChar == 'u' || currentChar == 'U'))

        {

            (\*head) = (\*head)->next;

            currentChar = (\*head)->data;

        }

*if* (head != *NULL*)

        {

            // *Iterate through the remaining nodes*

            Nodeptr temp = (\*head);

            Nodeptr prev = *NULL*;

*while* (temp != *NULL*)

            {

                currentChar = temp->data;

*if* (currentChar == 'a' || currentChar == 'A' || currentChar == 'e' || currentChar == 'E' || currentChar == 'i' || currentChar == 'I' || currentChar == 'o' || currentChar == 'O' || currentChar == 'u' || currentChar == 'U')

                {

                    prev->next = temp->next;

                }

*else*

                {

                    prev = temp;

                }

                temp = temp->next;

            }

        }

    }

}

// *prints the list*

void *printList*(Nodeptr head)

{

    Nodeptr temp = head;

*while* (temp->next != *NULL*)

    {

*printf*("%c->", temp->data);

        temp = temp->next;

    }

*if* (temp != *NULL*)

    {

*printf*("%c", temp->data);

    }

*printf*("\n");

}

int *main*()

{

    char \*s = "Suhan";

    Nodeptr head = *create*(s);

*printf*("The Entered Character is \n");

*printList*(head);

*allVowels*(&head);

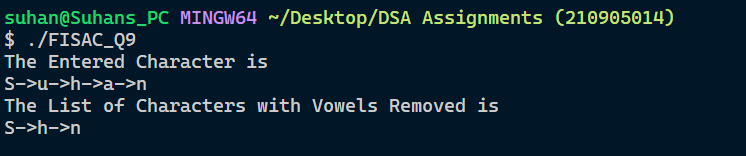
*printf*("The List of Characters with Vowels Removed is \n");

*printList*(head);

*return* 0;

}

**OUTPUT**

****

**Q10.**

#*include* <stdio.h>

#*include* <stdlib.h>

#*include* <string.h>

struct Node

{

    char \*data;

    int count;

    struct Node \*next;

};

void *displaycount*(struct Node \*first)

{

    struct Node \*temp = first;

*printf*("Contents of list are\n");

*while* (temp != *NULL*)

    {

*printf*("%s,%d\n", temp->data, temp->count);

        temp = temp->next;

    }

}

void *display*(struct Node \*first)

{

    struct Node \*temp = first;

*while* (temp != *NULL*)

    {

*printf*("%s\n", temp->data);

        temp = temp->next;

    }

}

struct Node \**insert\_rear*(char \*ele, struct Node \*first)

{

    struct Node \*ptr;

    ptr = (struct Node \*)*malloc*(sizeof(struct Node \*));

    ptr->data = (char \*)*malloc*(10 \* sizeof(char));

    ptr->data = ele;

*if* (first == *NULL*)

    {

        ptr->next = *NULL*;

*return* ptr;

    }

    struct Node \*temp = first;

*while* (temp->next != *NULL*)

    {

        temp = temp->next;

    }

    temp->next = ptr;

    ptr->next = *NULL*;

*return* first;

}

struct Node \**Create*()

{

    struct Node \*first;

    first = (struct Node \*)*malloc*(sizeof(struct Node \*));

    first = *NULL*;

*printf*("Enter number of students\n");

    int n, i;

*scanf*("%d", &n);

    char \*s1;

*printf*("Enter names of students\n");

*for* (i = 0; i < n; i++)

    {

        s1 = (char \*)*malloc*(10 \* sizeof(char));

*scanf*("%s", s1);

        first = *insert\_rear*(s1, first);

    }

*return* first;

}

struct Node \**delete\_rear*(struct Node \*first)

{

    struct Node \*temp1 = first;

    struct Node \*temp2 = first->next;

*while* (temp2->next != *NULL*)

    {

        temp2 = temp2->next;

        temp1 = temp1->next;

    }

    temp1->next = *NULL*;

*free*(temp2);

*return* first;

}

void *Search*(struct Node \*first)

{

    struct Node \*temp1 = first;

    struct Node \*temp3 = first;

*while* (temp1 != *NULL*)

    {

        struct Node \*temp2 = temp1->next;

        temp1->count = 1;

*while* (temp2 != *NULL*)

        {

*if* (*strcmp*(temp1->data, temp2->data) == 0 && temp2->next != *NULL*)

            {

                struct Node \*temp = temp2;

                temp3->next = temp2->next;

*free*(temp);

                temp2 = temp2->next;

                temp3 = temp3->next;

                temp1->count++;

            }

*else* *if* (*strcmp*(temp1->data, temp2->data) == 0 && temp2->next == *NULL*)

            {

                first = *delete\_rear*(first);

                temp1->count++;

                temp2 = temp2->next;

                temp3 = temp3->next;

            }

*else*

            {

                temp2 = temp2->next;

                temp3 = temp3->next;

            }

        }

        temp1 = temp1->next;

        temp3 = temp1;

    }

}

int *main*()

{

    struct Node \*head;

    head = (struct Node \*)*malloc*(sizeof(struct Node \*));

    head = *NULL*;

    head = *Create*();

*display*(head);

*printf*("After removing occurences :\n");

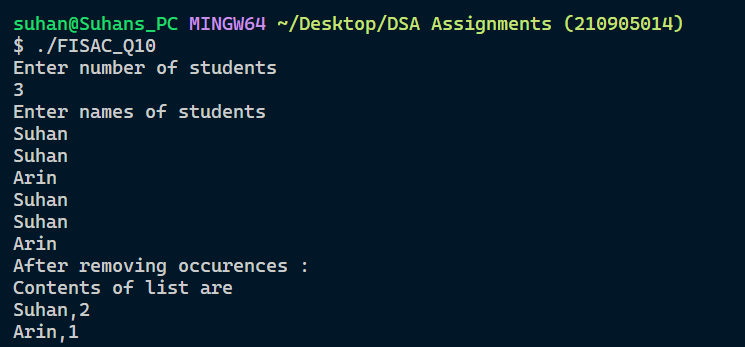
*Search*(head);

*displaycount*(head);

*return* 0;

}

**Output**

****