Assignment 4 – Linked List

Name: Kartik Banshi Katkar

Rollno : 29

Prn - 12111367

SY-IT-A

**Q.** **Implementation of Singly Linked List**

**Code:**

#include<stdio.h>

#include<stdlib.h>

struct node

{

    int data;

    struct node \*next;

};

struct node \*head;

void Insertbegin ();

void Insertend ();

void Insertrandom();

void begin\_delete();

void last\_delete();

void random\_delete();

void Print();

void search();

int main ()

{

    int choice =0;

    while(choice != 9)

    {

        printf("\n---------------------Menu---------------------\n");

        printf("\n1. Insert Element at the beginning of the Linked List");

        printf("\n2. Insert Element at end of Linked List");

        printf("\n3. Insert Element at any location in Linked List");

        printf("\n4. Delete Element from the beginning of Linked List");

        printf("\n5. Delete Element at End of Linked List");

        printf("\n6. Delete a node from the specified location");

        printf("\n7. Search for an Element in the Linked List");

        printf("\n8. Display the Elements");

        printf("\n9. exit");

        printf("\n\n Enter your choice : \t");

        scanf("\n%d",&choice);

        switch(choice)

        {

            case 1:

            Insertbegin();

            break;

            case 2:

            Insertend();

            break;

            case 3:

            Insertrandom();

            break;

            case 4:

            begin\_delete();

            break;

            case 5:

            last\_delete();

            break;

            case 6:

            random\_delete();

            break;

            case 7:

            search();

            break;

            case 8:

            Print();

            break;

            case 9:

            exit(0);

            break;

            default:

            printf("Please enter valid choice..");

        }

    }

}

void Insertbegin()

{

    struct node \*ptr;

    int item;

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

    if(ptr == NULL)

    {

        printf("\nOVERFLOW");

    }

    else

    {

        printf("\nEnter value to be Inserted\n");

        scanf("%d",&item);

        ptr->data = item;

        ptr->next = head;

        head = ptr;

        printf("\nNode inserted \n");

    }

}

void Insertend()

{

    struct node \*ptr,\*temp;

    int item;

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

    if(ptr == NULL)

    {

        printf("\nOVERFLOW");

    }

    else

    {

        printf("\nEnter value to be Inserted\n");

        scanf("%d",&item);

        ptr->data = item;

        if(head == NULL)

        {

            ptr -> next = NULL;

            head = ptr;

            printf("\nNode inserted \n");

        }

        else

        {

            temp = head;

            while (temp -> next != NULL)

            {

                temp = temp -> next;

            }

            temp->next = ptr;

            ptr->next = NULL;

            printf("\nNode inserted \n");

        }

    }

}

void Insertrandom()

{

    int i,loc,item;

    struct node \*ptr, \*temp;

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

    if(ptr == NULL)

    {

        printf("\nOVERFLOW");

    }

    else

    {

        printf("\nEnter element value");

        scanf("%d",&item);

        ptr->data = item;

        printf("\nEnter the location after which you want to insert: ");

        scanf("\n%d",&loc);

        temp=head;

        for(i=0;i<loc;i++)

        {

            temp = temp->next;

            if(temp == NULL)

            {

                printf("\nCannot be inserted\n");

                return;

            }

        }

        ptr ->next = temp ->next;

        temp ->next = ptr;

        printf("\nNode inserted Successfully");

    }

}

void begin\_delete()

{

    struct node \*ptr;

    if(head == NULL)

    {

        printf("\nList is empty\n");

    }

    else

    {

        ptr = head;

        head = ptr->next;

        free(ptr);

        printf("\nNode deleted from the beginning Successfully\n");

    }

}

void last\_delete()

{

    struct node \*ptr,\*ptr1;

    if(head == NULL)

    {

        printf("\nList is empty");

    }

    else if(head -> next == NULL)

    {

        head = NULL;

        free(head);

        printf("\nOnly node of the list deleted\n");

    }

    else

    {

        ptr = head;

        while(ptr->next != NULL)

        {

            ptr1 = ptr;

            ptr = ptr ->next;

        }

        ptr1->next = NULL;

        free(ptr);

        printf("\nDeleted Node from the last\n");

    }

}

void random\_delete()

{

    struct node \*ptr,\*ptr1;

    int loc,i;

    printf("\n Enter the location of the node after which you want to perform deletion \n");

    scanf("%d",&loc);

    ptr=head;

    for(i=0;i<loc;i++)

    {

        ptr1 = ptr;

        ptr = ptr->next;

        if(ptr == NULL)

        {

            printf("\nCannot Delete Element\n");

            return;

        }

    }

    ptr1 ->next = ptr ->next;

    free(ptr);

    printf("\nDeleted node %d ",loc+1);

}

void search()

{

    struct node \*ptr;

    int item,i=0,flag;

    ptr = head;

    if(ptr == NULL)

    {

        printf("\nEmpty List\n");

    }

    else

    {

        printf("\nEnter item which you want to search?\n");

        scanf("%d",&item);

        int yo=0;

        while (ptr!=NULL)

        {

            if(ptr->data == item)

            {

                printf("Item found at location %d ",i+1);

                yo = 0;

                break;

            }

            else

            {

                yo=1;

            }

            i++;

            ptr = ptr -> next;

        }

        if(yo==1)

        {

            printf("Item not found\n");

        }

    }

}

void Print()

{

    struct node \*ptr;

    ptr = head;

    if(ptr == NULL)

    {

        printf("Nothing to print");

    }

    else

    {

        printf("\nPrinting the values\n");

        while (ptr!=NULL)

        {

            printf("\n%d",ptr->data);

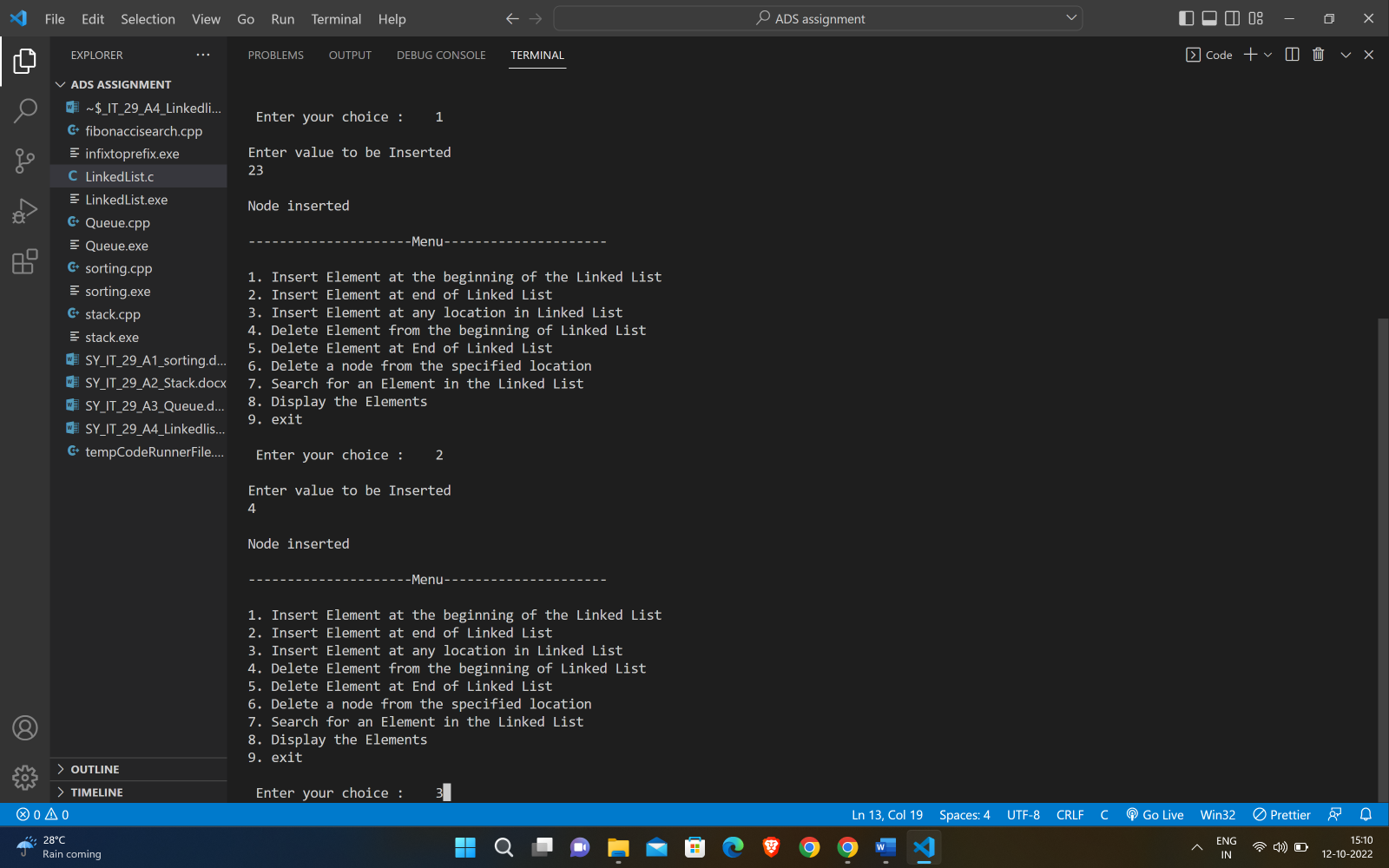
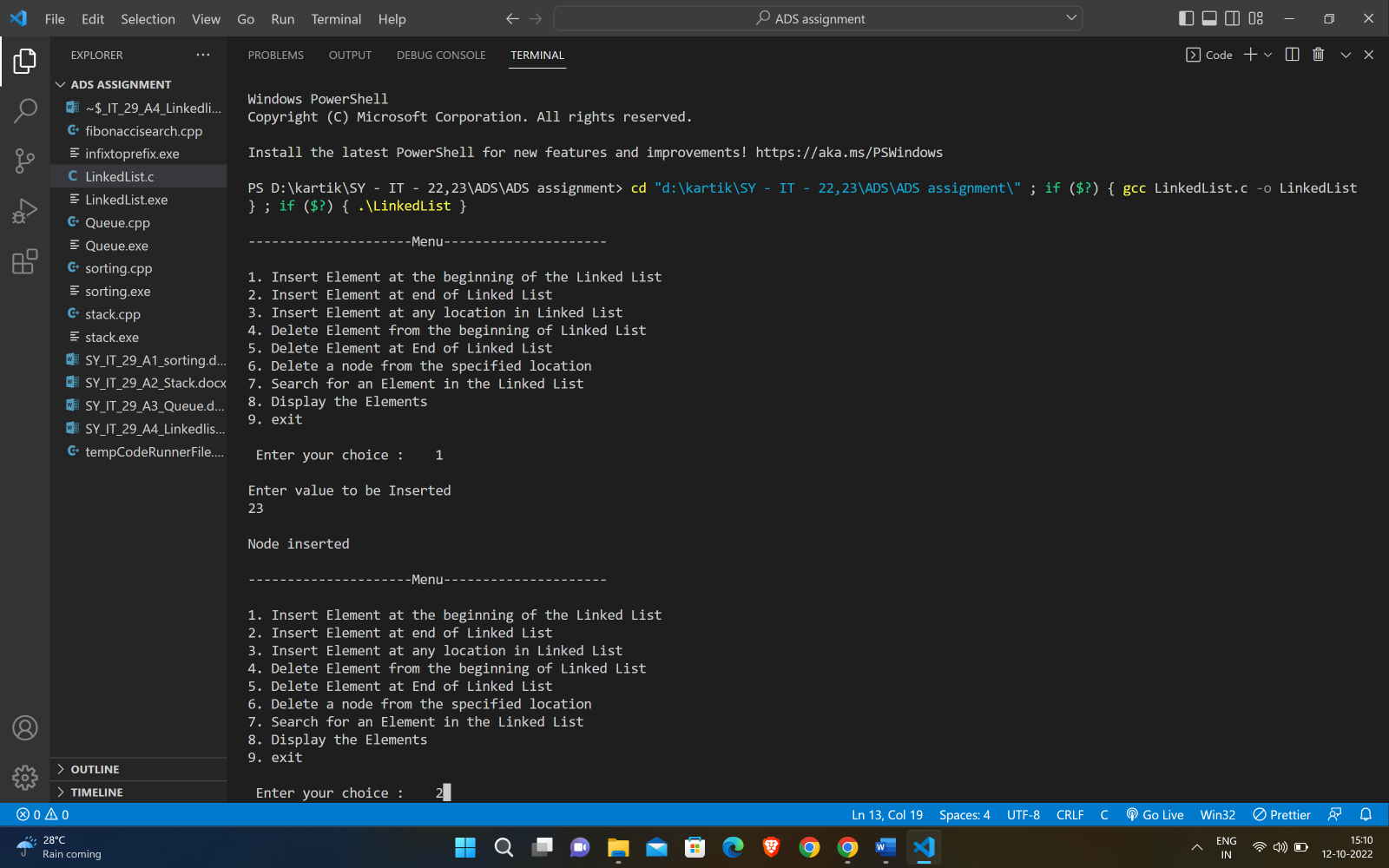
            ptr = ptr -> next;

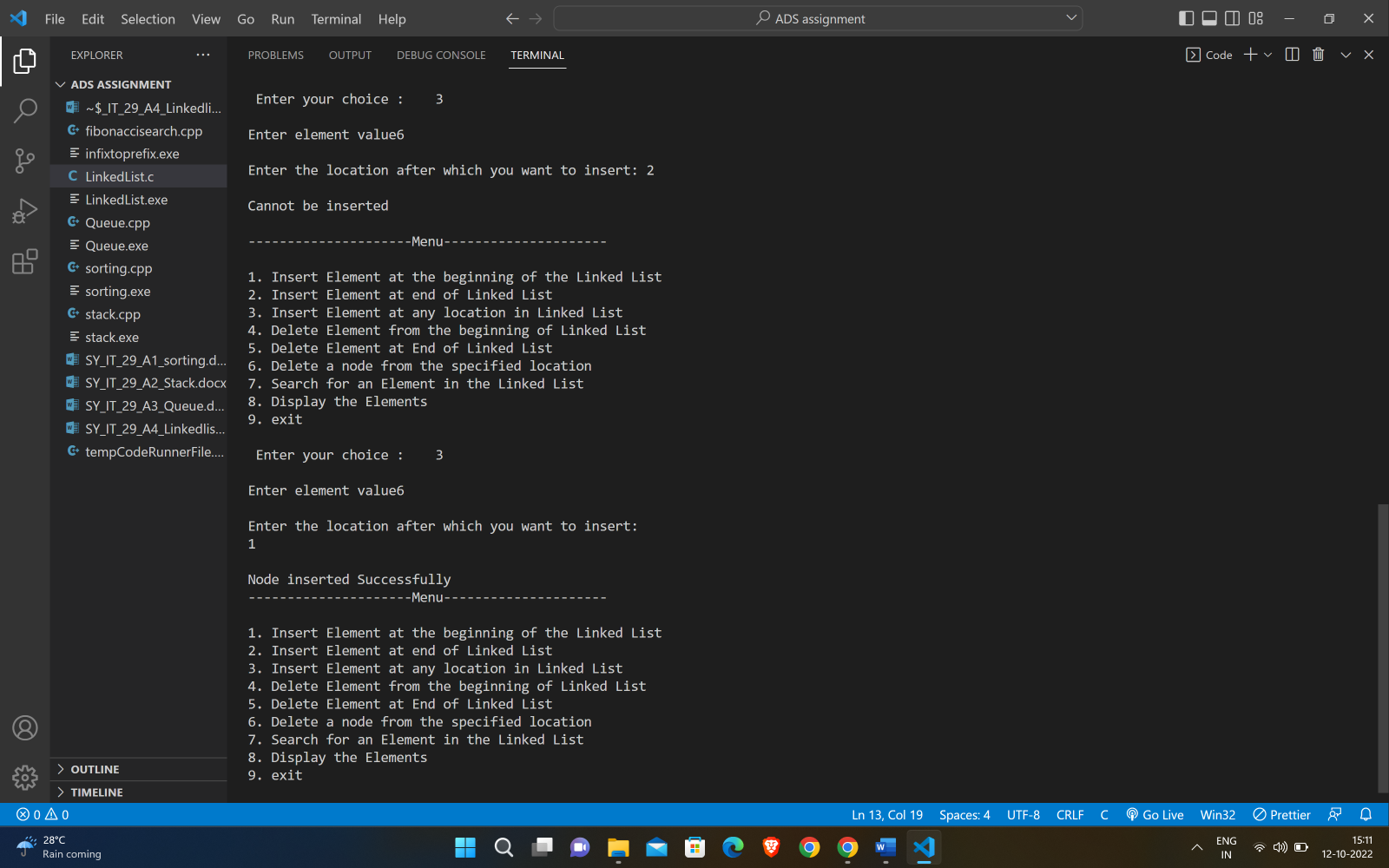
        }

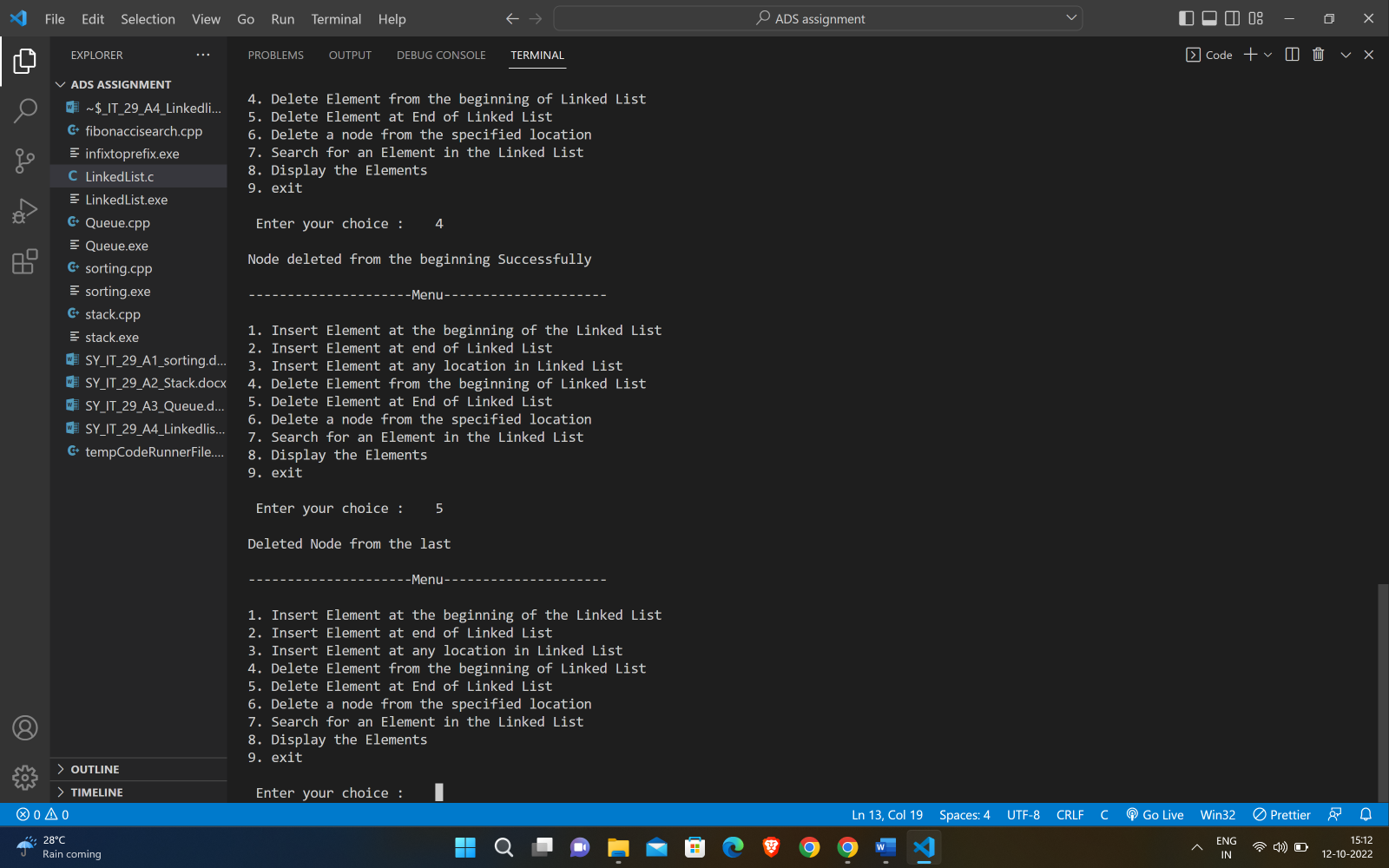
    }

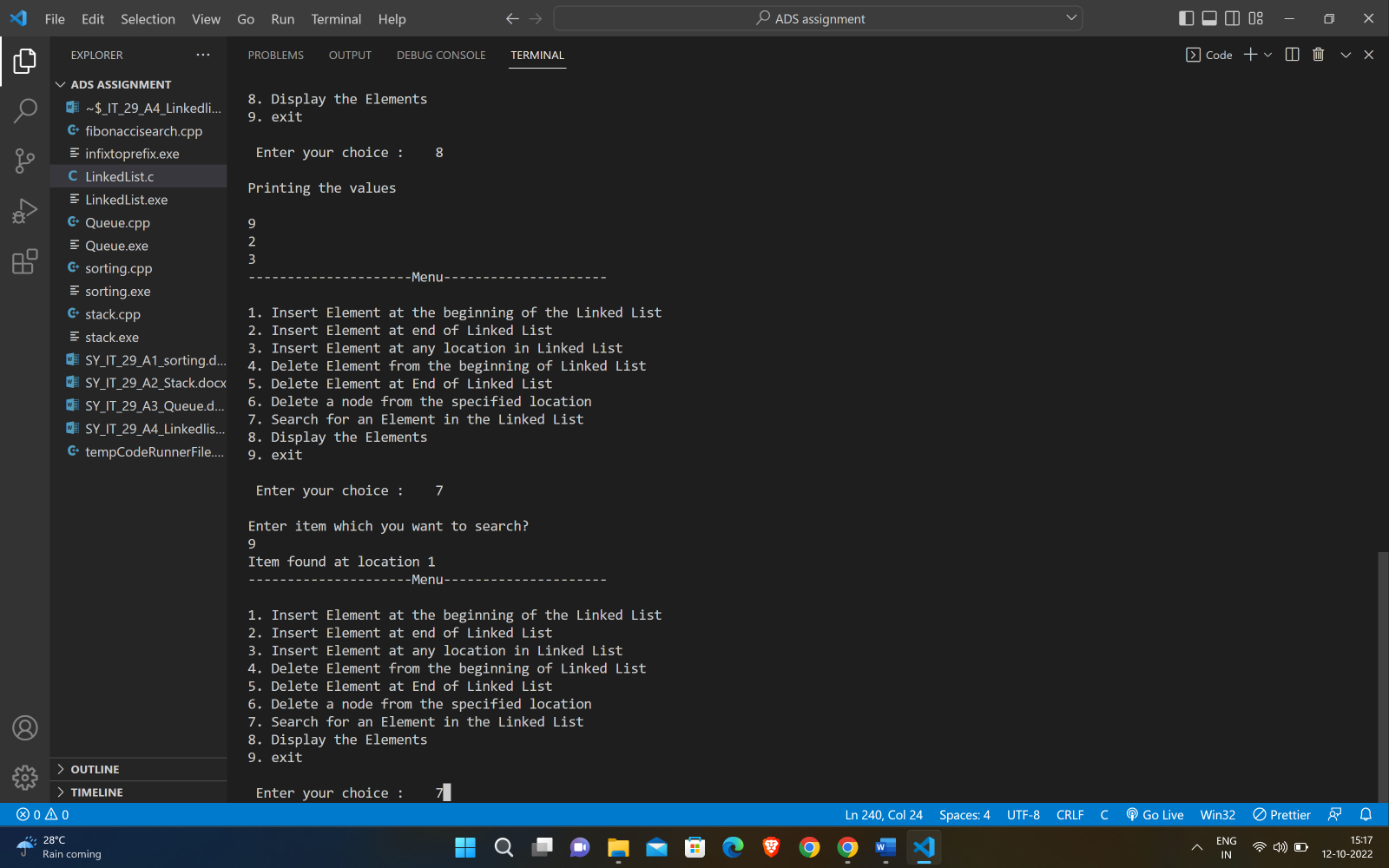
}

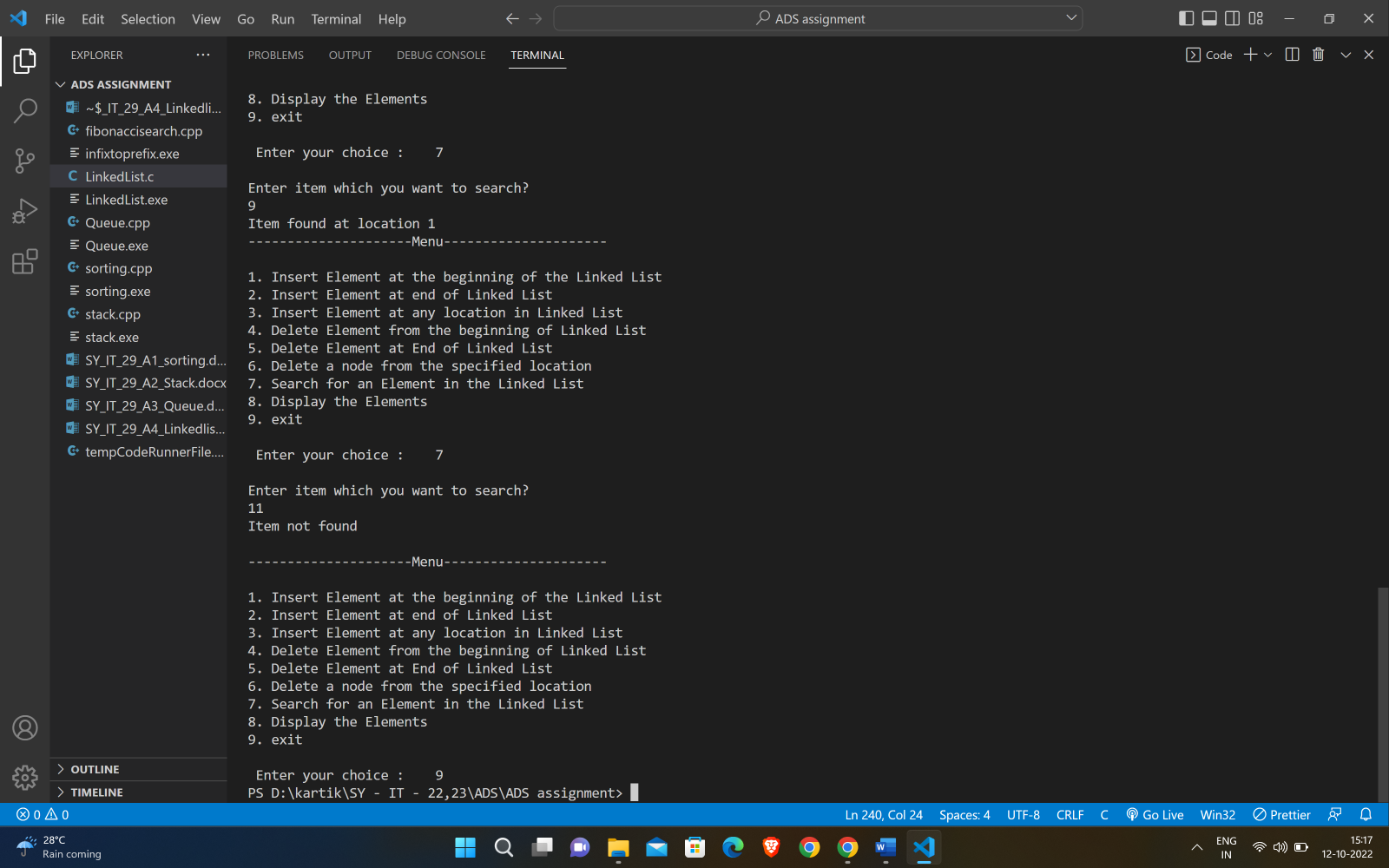
**Input / output:**

****

****

****

****

****