# Ds sample programs

1//single linked list opertations

#include <stdio.h>

#include <stdlib.h>

struct node

{

    int info;

    struct node \*next;

}\*start = NULL,\*current,\*new;

// struct node ;

void create()

{

    char ch;

    do

    {

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

        printf("\n enter info part: ");

        scanf("%d", &new->info);

        new->next = NULL;

        if (start == NULL)

        {

            current = new;

            start = new;

        }

        else

        {

            current->next = new;

            current = new;

        }

        printf("\n do you want to add another node: ");

        ch = getch();

    } while (ch == 'y');

}

void InsBeg()

{

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

    printf("\n what do you want to insert at beginning: ");

    scanf("%d", &new->info);

    new->next = start;

    start = new;

}

void Insend()

{

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

    printf("\n what do you want to insert at last: ");

    scanf("%d", &new->info);

    current = start;

    while (current->next != NULL)

    {

        current = current->next;

    }

    current->next = new;

    new->next = NULL;

}

void Delend()

{

    current = start;

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

    {

        current = current->next;

    }

    new = current->next;

    current->next = NULL;

    free(new);

}

void Delbeg()

{

    current = start;

    start = start->next;

    free(current);

}

void Traverse()

{

    current = start;

    while (current!=NULL)

    {

        printf("%d", current->info);

        if(current->next!=NULL){

            printf("-->");

        }

        current = current->next;

    }

}

// struct node \*start = 0;

int main()

{

    int ch;

    printf("\n what do you want to perform");

    while (1)

    {

        printf("\n Linked List Operation");

        printf("\n 1 Create");

        printf("\n 2 Insbeg");

        printf("\n 3 InsEnd");

        printf("\n 4 Delbeg");

        printf("\n 5 Delend");

        printf("\n 6 Traverse");

        printf("\n 7 Exit");

        printf("\nEnter Your Choice: ");

        scanf("%d", &ch);

        switch (ch)

        {

        case 1:

            create();

            break;

        case 2:

            InsBeg();

            break;

        case 3:

            Insend();

            break;

        case 4:

            Delbeg();

            break;

        case 5:

            Delend();

            break;

        case 6:

            Traverse();

            break;

        case 7:

            exit(12);

        default:

            printf("\n Please enter between (1-7)");

        }

    }

    return 0;

}

2//Stack Operations

#include <stdio.h>

#include <stdlib.h>

#define max 5

int stack[max];

int top = -1; // 0,1,2,3,4

void push()

{

    if (top == max - 1)

    {

        printf("\n overflow condition");

    }

    else

    {

        top++;

        int x;

        printf("\n what do yu wnt to enter: ");

        scanf("%d", &x);

        stack[top] = x;

    }

}

void pop()

{

    if (top == -1)

    {

        printf("\n underflow condition");

    }

    else

        printf("%d is popped", stack[top]);

    top--;

}

void peek()

{

    if (top == -1)

    {

        printf("\n underflow condition");

    }

    else

        printf("\n uppermost element is %d", stack[top]);

}

void isempty()

{

    if (top == -1)

    {

        printf("\n stack is empty");

    }

    else

        printf("\n stack is not empty");

}

void isFull()

{

    if (top == max - 1)

    {

        printf("\n fiul");

    }

    else

        printf("\n not full");

}

void display()

{

    if (top == -1)

    {

        printf("\n No elements to display");

    }

    else

    {

        printf("\n  elements are : ");

        for (int i = 0; i <= top; i++)

        {

            printf("%d ", stack[i]);

        }

    }

}

int main()

{

    int ch;

    top = -1;

    while (1)

    {

        printf("\n Stack Operation");

        printf("\n 1 Push");

        printf("\n 2 Pop");

        printf("\n 3 Peek");

        printf("\n 4 IsEmpty");

        printf("\n 5 IsFull");

        printf("\n 6 Display");

        printf("\n 7 Exit");

        printf("\nEnter Your Choice: ");

        scanf("%d", &ch);

        switch (ch)

        {

        case 1:

            push();

            break;

        case 2:

            pop();

            break;

        case 3:

            peek();

            break;

        case 4:

            isempty();

            break;

        case 5:

            isFull();

            break;

        case 6:

            display();

            break;

        case 7:

            exit(12);

        default:

            printf("\n Please enter between (1-7)");

        }

    }

}

//3Polynomials

// 6x2 + 5x + 3

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

struct node

{

    int co;

    int ex;

    struct node \*next;

};

struct node \*start1 = 0;

struct node \*start2 = 0;

struct node \*new1, \*new2, \*current1, \*current2;

void create()

{

    char ch;

    printf("\n 1st Polynomial");

    do

    {

        new1 = (struct node \*)malloc(sizeof(struct node));

        printf("\n enter the coefficient");

        scanf("%d", &new1->co);

        printf("\n enter the exponent ");

        scanf("%d", &new1->ex);

        new1->next = 0;

        if (start1 == 0)

        {

            current1 = new1;

            start1 = new1;

        }

        else

        {

            current1->next = new1;

            current1 = new1;

        }

        printf("\n want to add more??");

        ch = getch();

    } while (ch == 'y');

    printf("\n 1st Polynomial");

    do

    {

        new2 = (struct node \*)malloc(sizeof(struct node));

        printf("\n enter the coefficient");

        scanf("%d", &new2->co);

        printf("\n enter the exponent ");

        scanf("%d", &new2->ex);

        new2->next = 0;

        if (start2 == 0)

        {

            current2 = new2;

            start2 = new2;

        }

        else

        {

            current2->next = new2;

            current2 = new2;

        }

        printf("\n want to add more??");

        ch = getch();

    } while (ch == 'y');

}

void display()

{

    printf("\n your first polynomial is ");

    current1 = start1;

    while (current1 != 0)

    {

        printf("%dx^%d", current1->co, current1->ex);

        current1 = current1->next;

        if (current1 != 0)

        {

            printf(" + ");

        }

    }

    printf("\n");

    printf("your second polynomial is ");

    current2 = start2;

    while (current2 != 0)

    {

        printf("%dx^%d", current2->co, current2->ex);

        current2 = current2->next;

        if (current2 != 0)

        {

            printf(" + ");

        }

    }

}

void add()

{

    current1 = start1;

    current2 = start2;

    printf("\n");

    while (current1 != 0 && current2 != 0)

    {

        if (current1->ex == current2->ex)

        {

            printf("%dx^%d", current1->co + current2->co, current1->ex);

        }

        if (current1->ex == current2->ex)

        {

        }

        if (current1->ex == current2->ex)

        {

        }

    }

}

int main()

{

    int i = 1;

    while (i)

    {

        printf("\n what do u want to perform");

        printf("1.creation\n 2.Display \n 3.Addition");

        int ch;

        scanf("%d",&ch);

        switch (ch)

        {

        case 1:

            create();

            break;

        default:

            break;

        }

    }

    create();

    display();

    return 0;

}

//04Queue Operations

//Linear Queue

#define max 4

#include <stdio.h>

#include <stdlib.h>

int Queue[max];

int front = -1;

int rear = -1;

void enqueue();

{

    if (rear == max - 1)

    {

        printf("\noverflow Condition");

    }

    else

    {

        if (front == -1)

        {

            front = 0;

        }

        rear += 1;

        int x;

        printf("\n what do you want to add");

        scanf("%d", &x);

        Queue[rear] = x;

    }

}

void dequeue()

{

    if (rear == -1)

        printf("\nno elements to perform Dequeue ");

    else

    {

        int y;

        y = Queue[front];

        if (front == rear)

        {

            front = -1;

            rear = -1;

        }

        else

        {

            front++;

        }

        printf("%d is removed", y);

    }

}

void traverse()

{

    if (rear == -1)

        printf("\nno elements to Traverse ");

    else

    {

        int i;

        printf("\n Elements in queue are: ");

        for (i = front; i <= rear; i++)

        {

            printf("%d ", Queue[i]);

        }

    }

}

void peek()

{

    printf("\n last inserted element is %d and first inserted element is %d", Queue[rear], Queue[front]);

}

void IsFull()

{

    if (rear == max-1)

        printf("\nQueue is full");

    else

        printf("\nQueue is Not full");

}

void IsEmpty()

{

    if (rear == -1)

        printf("\nQueue is empty");

    else

        printf("\nQueue is Not empty");

}

int main()

{

    int ch;

    printf("\n what do you want to perform");

    while (1)

    {

        printf("\n Stack Operation");

        printf("\n 1 Enqueue");

        printf("\n 2 Dequeu");

        printf("\n 3 traverse");

        printf("\n 4 peek");

        printf("\n 5 IsFull");

        printf("\n 6 IsEmpty");

        printf("\n 7 Exit");

        printf("\nEnter Your Choice: ");

        scanf("%d", &ch);

        switch (ch)

        {

        case 1:

            enqueue();

            break;

        case 2:

            dequeue();

            break;

        case 3:

            traverse();

            break;

        case 4:

            peek();

            break;

        case 5:

            IsFull();

            break;

        case 6:

            IsEmpty();

            break;

        case 7:

            exit(12);

        default:

            printf("\n Please enter between (1-7)");

        }

    }

}

//Circular Queue

#define max 4

#include <stdio.h>

#include <stdlib.h>

int queue[max];

int front = -1;

int rear = -1;

void enqueue()

{

    if ((front == 0) && (rear == max - 1) || ((front - rear) == 1))

    {

        printf("\nqueue is full");

    }

    else

    {

        int nu;

        if (front == -1)

        {

            front = 0;

            rear = 0;

        }

        else if (rear == max - 1)

        {

            rear = 0;

        }

        else

        {

            rear++;

        }

        printf("\n what do u want to enter? ");

        scanf("%d", &nu);

        queue[rear] = nu;

    }

}

void dequeue()

{

    if (front == -1)

    {

        printf("\nno elements to display");

    }

    else

    {

        int nr = queue[front];

        printf("\n %d popped", nr);

        if (front == rear)

        {

            front = -1;

            rear = -1;

        }

        else if (front == max - 1)

        {

            front = 0;

        }

        else

        {

            front++;

        }

    }

}

void traverse()

{

    if (front == -1)

    {

        printf("\nno elements to display");

    }

    else

    {

        printf("\n Elements are");

        if (front > rear)

        {

            for (int i = front; i < max; i++)

            {

                printf("\t%d", queue[i]);

            }

            for (int i = 0; i <= rear; i++)

            {

                printf("\t%d", queue[i]);

            }

        }

        else

        {

            for (int i = front; i <= rear; i++)

            {

                printf("\t%d", queue[i]);

            }

        }

    }

}

void peek()

{

    printf("\n last inserted element is %d and first inserted element is %d", queue[rear], queue[front]);

}

void IsFull()

{

    if ((front == 0) && (rear == max - 1) || ((front - rear) == 1))

        printf("\nQueue is full");

    else

        printf("\nQueue is Not full");

}

void IsEmpty()

{

    if (rear == -1)

        printf("\nQueue is empty");

    else

        printf("\nQueue is Not empty");

}

int main()

{

    int ch;

    while (1)

    {

    printf("\n what do you want to perform");

        printf(" from following Stack Operation");

        printf("\n 1 Enqueue");

        printf("\n 2 Dequeu");

        printf("\n 3 traverse");

        printf("\n 4 peek");

        printf("\n 5 IsFull");

        printf("\n 6 IsEmpty");

        printf("\n 7 Exit");

        printf("\n Enter Your Choice: ");

        scanf("%d", &ch);

        switch (ch)

        {

        case 1:

            enqueue();

            break;

        case 2:

            dequeue();

            break;

        case 3:

            traverse();

            break;

        case 4:

            peek();

            break;

        case 5:

            IsFull();

            break;

        case 6:

            IsEmpty();

            break;

        case 7:

            exit(12);

        default:

            printf("\n Please enter between (1-7)");

        }

    }

}

#include <stdio.h>

int smallest(int arr[], int j, int size)

{

    int s = j;

    for (int i = j; i < size; i++)

    {

        if (arr[i] < s)

        {

            s = i;

        }

    }

    return s;

}

void swap(int \*n1, int \*n2)

{

    \*n1 = \*n1 + \*n2;

    \*n2 = \*n1 - \*n2;

    \*n1 = \*n1 - \*n2;

}

void SelectionSort(int arr[], int size)

{

    for (int i = 0; i < size; i++)

    {

        int min = smallest(arr, i, size);

        if (min != i)

        {

            swap(&arr[min], &arr[i]);

        }

    }

    for (int i = 0; i < size; i++)

    {

        printf("%d ", arr[i]);

    }

}

void InsertionSort(int arr[], int size)

{

    int key, j;

    for (int i = 1; i < size; i++)

    {

        key = arr[i];

        j = i - 1;

        while (j >= 0 && arr[j] > key)

        {

            arr[j + 1] = arr[j];

            j--;

        }

        arr[j + 1] = key;

    }

    for (int i = 0; i < size; i++)

    {

        printf("%d ", arr[i]);

    }

}

void BubbleSort(int arr[], int size)

{

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

    {

        for (int j = 0; j < size - i - 1; i++)

        {

            if (arr[j] > arr[j + 1])

                swap(&arr[j], &arr[j + 1]);

        }

    }

    for (int i = 0; i < size; i++)

    {

        printf("%d ", arr[i]);

    }

}

void linear(int arr[], int size, int key)

{

    for (int i = 0; i < size; i++)

    {

        if (arr[i] == key)

        {

            printf("%d is present at %d position", key, i + 1);

            break;

        }

    }

}

void binary(int arr[], int size, int key)

{

    int l = 0;

    int h = size;

    while (l <= h)

    {

        int mid = (l + (h)) / 2;

        if (arr[mid] == key)

        {

            printf("%d is present at %d position ", key, mid+1);

            return;

        }

        else if (arr[mid] < key)

        {

            l = mid + 1;

        }

        else

            h = mid - 1;

    }

    printf("%d is not present ", key);

}

int main()

{

    int arr[100];

    int size;

    printf("\n how many elements you want to enter");

    scanf("%d", &size)

        printf("\n Enter elements one by one");

    for (int i = 0; i < size; i++)

    {

        scanf("%d " & arr[i]);

    }

    SelectionSort(arr, size);

    InsertionSort(arr, size);

    BubbleSort(arr, size);

    binary(arr,5,7);

    return 0;

}