14th may

File handling programs

1.

```
rps@rps-virtual-machine:~$ vi filehandling.c
rps@rps-virtual-machine:~$ gcc filehandling.c
rps@rps-virtual-machine:~$ ./a.out
Enter number::99
rps@rps-virtual-machine:~$ cat filehandling.c
#include <stdio.h>
#include <stdlib.h>
void main ()
{


    int n;
    FILE *fptr;
    fptr = fopen ("C:\\Downloads\\testtext.txt", "a");
    if (fptr == NULL)
    {
        printf ("Error!!!!!");
        exit(0);



    }
    printf ("Enter number::");
    scanf ("%d", &n);


    fprintf (fptr, "%d", n);
    fclose (fptr);
}
rps@rps-virtual-machine:~$ gcc filehandling.c
rps@rps-virtual-machine:~$ ./a.out
Enter number::100
rps@rps-virtual-machine:~$
```

Output

```
C:\Downloads\testtext.txt
99100
```

2.employee management system

```
#include <stdio.h>
#include <string.h>
void create();
```

```
void view();
void edit();
void delete();
void choice();


void choice(){
    int choic;
    printf("\nWELCOME TO OMG COMPANY\n");
    start:
    printf("--------------------------------\n");
    printf("Press 1 to Create a new Data\n");
    printf("Press 2 to View the Data\n");
    printf("Press 3 to edit the Data\n");
    printf("Press 4 to delete the Data\n");
    printf("Press 5 to exit the Program...!\n");
    printf("--------------------------------\n");

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

    if(choice==5){
        printf("Exiting the Program...\n");
        return ;
    }
    switch (choice)
    {
    case 1:
        create();
        printf("Do you wish to continue? 1 for Yes 0 for No\n");
        scanf("%d",&choic);

        if(choic==1){
            goto start;
        }
        else if(choic==0){
            printf(" exiting the program......!\n");
        }
        break;
    case 2:
        view();
        printf("Do you wish to continue? 1 for Yes 0 for No\n");
        scanf("%d",&choic);

        if(choic==1){
            goto start;
        }
        else if(choic==0){
```

```
            printf(" exiting the program......!\n");
        }
        break;
    case 3:
        edit();
        printf("Do you wish to continue? 1 for Yes 0 for No\n");
        scanf("%d",&choic);

        if(choic==1){
            goto start;
        }
        else if(choic==0){
            printf(" exiting the program......!\n");
        }
        break;
    case 4:
        delete();
        printf("Do you wish to continue? 1 for Yes 0 for No\n");
        scanf("%d",&choic);

        if(choic==1){
            goto start;
        }
        else if(choic==0){
            printf(" exiting the program......!\n");
        }
        break;
    default:
        break;
    }
}
struct emp
{
    int id;
    char name[30];
    int age;
    char branch[20];
    char designation[20];
}emp;
void create(){
    struct emp p1;
    FILE *fp=fopen("D:\\Assignments\\Demo1.txt", "a");
    if(fp==NULL){
        printf("Error opening File");
        return ;
    }
    printf("\nEnter Employee Id:");
    scanf("%d",&p1.id);
```

```
    // fflush(stdin);

    printf("\nEnter Employee name: ");
    scanf("%s", p1.name);
    strcpy (p1.name, p1.name);

    printf("\nEnter Employee Age: ");
    scanf("%d", &p1.age);

    printf("\nEnter Employee Branch:");
    scanf("%s", p1.branch);

    printf("\nEnter Employee Designation:");
    scanf("%s", p1.designation);


    //fprintf(fp,"Employee Id:%d\n Employee Name:%s\n Employee Age: %d \n Employee
Branch:%s\n Employee Designation :%s\n\n
",p1.id,p1.name,p1.age,p1.branch,p1.designation);
    fwrite(&p1,sizeof(emp),1,fp);
    printf("Data Uploaded Successfully\n");
    fclose(fp);
}

void view(){
    struct emp p1;
    FILE *ptr;
    ptr=fopen("D:\\Assignments\\Demo1.txt", "r");
    if(ptr==NULL){
        printf("Failed to open File\n");
        return;
    }
    while(fread(&p1,sizeof(emp),1,ptr)){
        printf("%d %s %d %s %s\n",p1.id,p1.name,p1.age,p1.branch,p1.designation);
    }
    fclose(ptr);
}

void edit(){
     struct emp p1;
    FILE *ptr, *ptr1;
    int c = 0;
    int a;

    ptr = fopen("D:\\Assignments\\Demo1.txt", "r");
    if (ptr == NULL) {
        printf("Error opening file for reading\n");
        return ;
```

```
}

ptr1 = fopen("D:\\Assignments\\temp.txt", "w");
if (ptr1 == NULL) {
    printf("Error opening file for writing\n");
    fclose(ptr);
    return ;
}

printf("Enter the Employee Id to Modify the Data:\n");
scanf("%d", &a);

while (fread(&p1, sizeof(struct emp), 1, ptr)) {
    if (a == p1.id) {
        c = 1;
        printf("Enter New Employee name: \n");
        scanf("%s", p1.name);

        printf("Enter New Employee Age: \n");
        scanf("%d", &p1.age);

        printf("Enter New Employee Branch:\n");
        scanf("%s", p1.branch);

        printf("Enter New Employee Designation:\n");
        scanf("%s", p1.designation);
    }
    fwrite(&p1, sizeof(struct emp), 1, ptr1);
}

fclose(ptr);
fclose(ptr1);

if (c == 0) {
    printf("Employee Not found\n");
} else {
    ptr1 = fopen("D:\\Assignments\\temp.txt", "r");
    ptr = fopen("D:\\Assignments\\Demo1.txt", "w");
    if (ptr == NULL) {
        printf("Error opening file for writing\n");
        fclose(ptr1);
        return ;
    }
    if (ptr1 == NULL) {
        printf("Error opening file for reading\n");
        fclose(ptr);
        return ;
    }
```

```
        while (fread(&p1, sizeof(struct emp), 1, ptr1)) {
            fwrite(&p1, sizeof(struct emp), 1, ptr);
        }

        fclose(ptr);
        fclose(ptr1);
        remove("D:\\Assignments\\temp.txt");
    }

}

void delete(){
    FILE *ptr, *ptr1;
    struct emp p1;

    ptr = fopen("D:\\Assignments\\Demo.txt", "r");
    ptr1 = fopen("D:\\Assignments\\temp.txt", "w");

    if(ptr == NULL){
        printf("Error opening files for deleting\n");
        fclose(ptr);

        return;
    }
    if(ptr1==NULL){
        printf("Error opening files for deleting\n");
        fclose(ptr1);
        return ;
    }
    printf("Enter the employee id to delete\n");
    int a;
    scanf("%d", &a);
    int c = 0;
    while (fread(&p1, sizeof(struct emp), 1, ptr)){
        printf("%d",p1.id);
        if(a == p1.id){
            c = 1;
        } else {
            fwrite(&p1, sizeof(struct emp), 1, ptr1);
        }
    }
    fclose(ptr1);
    fclose(ptr);

    if(c == 1){
        ptr = fopen("D:\\Assignments\\temp.txt", "r");
        ptr1 = fopen("D:\\Assignments\\Demo1.txt", "w");
```

```
        if(ptr == NULL || ptr1 == NULL){
            printf("Failed to open the file\n");
            if(ptr) fclose(ptr);
            if(ptr1) fclose(ptr1);
            return;
        }

        while(fread(&p1, sizeof(struct emp), 1, ptr)){
            fwrite(&p1, sizeof(struct emp), 1, ptr1);
        }
        fclose(ptr1);
        fclose(ptr);
        remove("D:\\Assignments\\temp.txt");
        printf("Deleted the employee successfully\n");
    } else {
        printf("Record not found\n");
    }
}
int main()
{
    choice();
    return 0;
}
```

```
Press 2 to View the Data
Press 3 to edit the Data
PS C:\Users\T.Aruna Reddy\Desktop\C programs> gcc emp.c
PS C:\Users\T.Aruna Reddy\Desktop\C programs> .\a.exe

WELCOME TO OMG COMPANY
-------------------------------
Press 1 to Create a new Data
Press 2 to View the Data
Press 3 to edit the Data
Press 4 to delete the Data
Press 5 to exit the Program...!
-------------------------------
```

15th may

1.program to print the calender for one year

```
#include <stdio.h>
```

```
// Function that returns the index of the
// day for date DD/MM/YYYY
int dayNumber(int day, int month, int year)
{

    static int t[] = { 0, 3, 2, 5, 0, 3,
                       5, 1, 4, 6, 2, 4 };
    year -= month < 3;
    return (year + year / 4
            - year / 100
            + year / 400
            + t[month - 1] + day)
        % 7;
}

// Function that returns the name of the
// month for the given month Number
// January - 0, February - 1 and so on
char* getMonthName(int monthNumber)
{
    char* month;

    switch (monthNumber) {
    case 0:
        month = "January";
        break;
    case 1:
        month = "February";
        break;
    case 2:
        month = "March";
        break;
    case 3:
        month = "April";
        break;
    case 4:
        month = "May";
        break;
    case 5:
        month = "June";
        break;
    case 6:
        month = "July";
        break;
    case 7:
        month = "August";
        break;
```

```
    case 8:
        month = "September";
        break;
    case 9:
        month = "October";
        break;
    case 10:
        month = "November";
        break;
    case 11:
        month = "December";
        break;
    }
    return month;
}

// Function to return the number of days
// in a month
int numberOfDays(int monthNumber, int year)
{
    // January
    if (monthNumber == 0)
        return (31);

    // February
    if (monthNumber == 1) {
        // If the year is leap then Feb
        // has 29 days
        if (year % 400 == 0
            || (year % 4 == 0
                && year % 100 != 0))
            return (29);
        else
            return (28);
    }

    // March
    if (monthNumber == 2)
        return (31);

    // April
    if (monthNumber == 3)
        return (30);

    // May
    if (monthNumber == 4)
        return (31);
```

```
    // June
    if (monthNumber == 5)
        return (30);

    // July
    if (monthNumber == 6)
        return (31);

    // August
    if (monthNumber == 7)
        return (31);

    // September
    if (monthNumber == 8)
        return (30);

    // October
    if (monthNumber == 9)
        return (31);

    // November
    if (monthNumber == 10)
        return (30);

    // December
    if (monthNumber == 11)
        return (31);
}

// Function to print the calendar of
// the given year
void printCalendar(int year)
{
    printf("     Calendar - %d\n\n", year);
    int days;

    // Index of the day from 0 to 6
    int current = dayNumber(1, 1, year);

    // i for Iterate through months
    // j for Iterate through days
    // of the month - i
    for (int i = 0; i < 12; i++) {
        days = numberOfDays(i, year);

        // Print the current month name
        printf("\n ------------%s-------------\n",
            getMonthName(i));
```

```
        // Print the columns
        printf(" Sun Mon Tue Wed Thu Fri Sat\n");

        // Print appropriate spaces
        int k;
        for (k = 0; k < current; k++)
            printf("     ");

        for (int j = 1; j <= days; j++) {
            printf("%5d", j);

            if (++k > 6) {
                k = 0;
                printf("\n");
            }
        }

        if (k)
            printf("\n");

        current = k;
    }

    return;
}

// Driver Code
int main()
{
    int year = 2016;

    // Function Call
    printCalendar(year);
    return 0;
}
```

output:

```
PS C:\Users\T.Aruna Reddy\Desktop\C programs> gcc initial.c
PS C:\Users\T.Aruna Reddy\Desktop\C programs> .\a.exe
     Calendar - 2016


 ------------January-------------
 Sun Mon Tue Wed Thu Fri Sat
                         1    2
    3    4    5    6    7    8    9
   10   11   12   13   14   15   16
```

17 18 19 20 21 22 23
24 25 26 27 28 29 30
31

----------February----------

Sun Mon Tue Wed Thu Fri Sat
1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29

----------March----------

Sun Mon Tue Wed Thu Fri Sat
1 2 3 4 5
6 7 8 9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31

----------April----------

Sun Mon Tue Wed Thu Fri Sat
1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30

----------May----------

Sun Mon Tue Wed Thu Fri Sat
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31

----------June----------

Sun Mon Tue Wed Thu Fri Sat
1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30

----------July----------

Sun Mon Tue Wed Thu Fri Sat
1 2
3 4 5 6 7 8 9

```
 10   11   12   13   14   15   16
 17   18   19   20   21   22   23
 24   25   26   27   28   29   30
 31

------------August-------------
Sun Mon Tue Wed Thu Fri Sat
       1    2    3    4    5    6
  7    8    9   10   11   12   13
 14   15   16   17   18   19   20
 21   22   23   24   25   26   27
 28   29   30   31

------------September-------------
Sun Mon Tue Wed Thu Fri Sat
                  1    2    3
  4    5    6    7    8    9   10
 11   12   13   14   15   16   17
 18   19   20   21   22   23   24
 25   26   27   28   29   30

------------October-------------
Sun Mon Tue Wed Thu Fri Sat
                         1
  2    3    4    5    6    7    8
  9   10   11   12   13   14   15
 16   17   18   19   20   21   22
 23   24   25   26   27   28   29
 30   31

------------November-------------
Sun Mon Tue Wed Thu Fri Sat
           1    2    3    4    5
  6    7    8    9   10   11   12
 13   14   15   16   17   18   19
 20   21   22   23   24   25   26
 27   28   29   30

------------December-------------
Sun Mon Tue Wed Thu Fri Sat
                  1    2    3
  4    5    6    7    8    9   10
 11   12   13   14   15   16   17
 18   19   20   21   22   23   24
 25   26   27   28   29   30   31
```

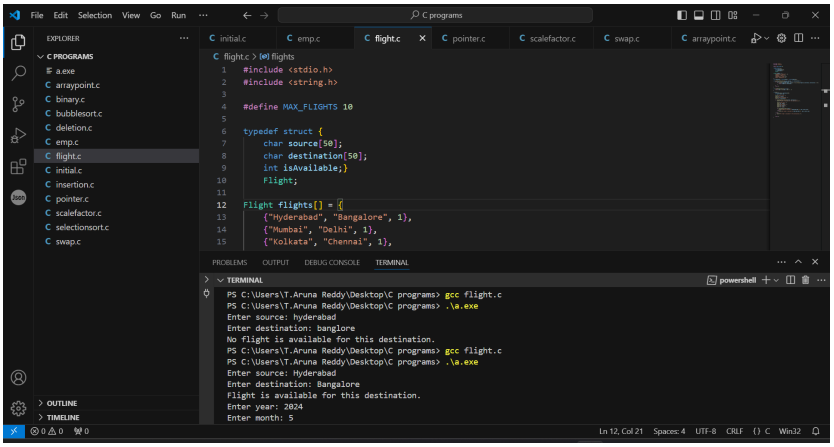
2.flight management system project:

```
#include <stdio.h>
```

```
#include <string.h>

#define MAX_FLIGHTS 10

typedef struct {
    char source[50];
    char destination[50];
    int isAvailable;}
    Flight;

Flight flights[] = {
    {"Hyderabad", "Bangalore", 1},
    {"Mumbai", "Delhi", 1},
    {"Kolkata", "Chennai", 1},
    {"Bhubaneswar", "Pune", 0},
    {"Goa", "Chandigarh", 0}
};
int flightCount = sizeof(flights) / sizeof(flights[0]);

int checkFlightAvailability(char source[], char destination[]) {
    for (int i = 0; i < flightCount; i++) {
        if (strcmp(flights[i].source, source) == 0 && strcmp(flights[i].destination, destination) ==
0) {
            return flights[i].isAvailable;
        }
    }
    return 0;
}

int checkDateAvailability(int day) {
    return (day >= 1 && day <= 20) ? 1 : 0;
}

int main() {
    char source[50], destination[50];
    int year, month, day;

    printf("Enter source: ");
    scanf("%s", source);
    printf("Enter destination: ");
    scanf("%s", destination);

    if (checkFlightAvailability(source, destination)) {
        printf("Flight is available for this destination.\n");

        printf("Enter year: ");
        scanf("%d", &year);
        printf("Enter month: ");
```

```
        scanf("%d", &month);
        printf("Enter day: ");
        scanf("%d", &day);

        if (checkDateAvailability(day)) {
            printf("Flight is available on %04d-%02d-%02d.\n", year, month, day);
        } else {
            printf("No flight is available on %04d-%02d-%02d.\n", year, month, day);
        }
    } else {
        printf("No flight is available for this destination.\n");
    }

    return 0;
}
```

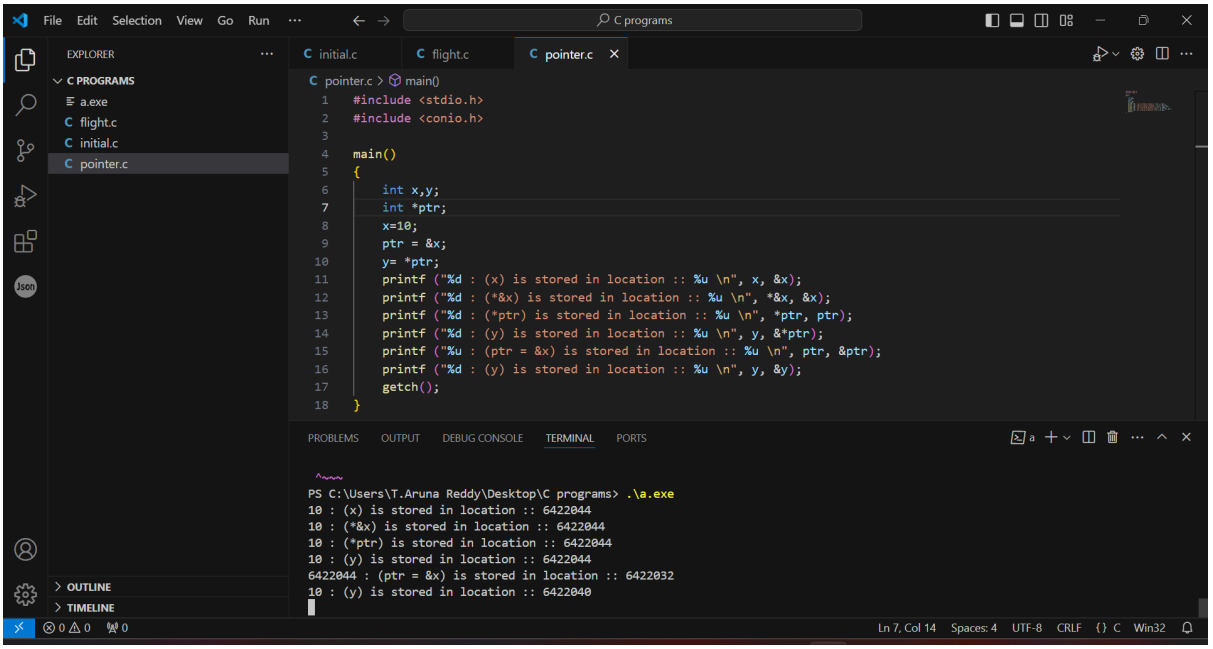
Output:



16th may

1.pointer

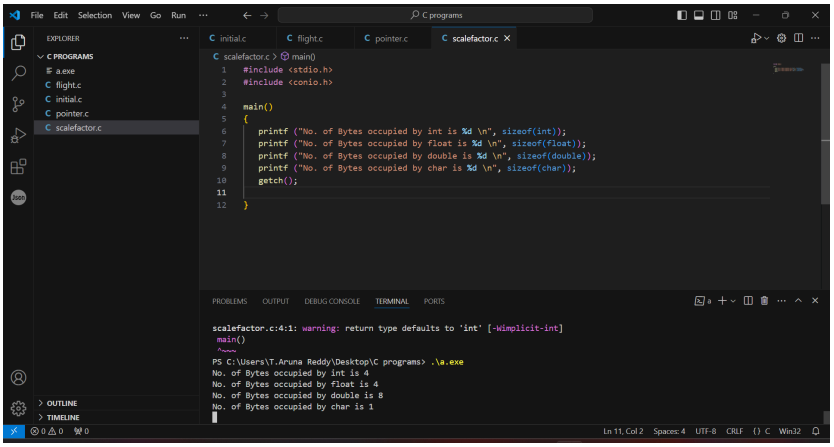
```c
#include <stdio.h>
#include <conio.h>

main()
{
   int x,y;
   int *ptr;
   x=10;
   ptr = &x;
   y= *ptr;
   printf ("%d : (x) is stored in location :: %u \n", x, &x);
   printf ("%d : (*&x) is stored in location :: %u \n", *&x, &x);
   printf ("%d : (*ptr) is stored in location :: %u \n", *ptr, ptr);
   printf ("%d : (y) is stored in location :: %u \n", y, &*ptr);
   printf ("%u : (ptr = &x) is stored in location :: %u \n", ptr, &ptr);
   printf ("%d : (y) is stored in location :: %u \n", y, &y);
   getch();
}
```



2.Scale factor

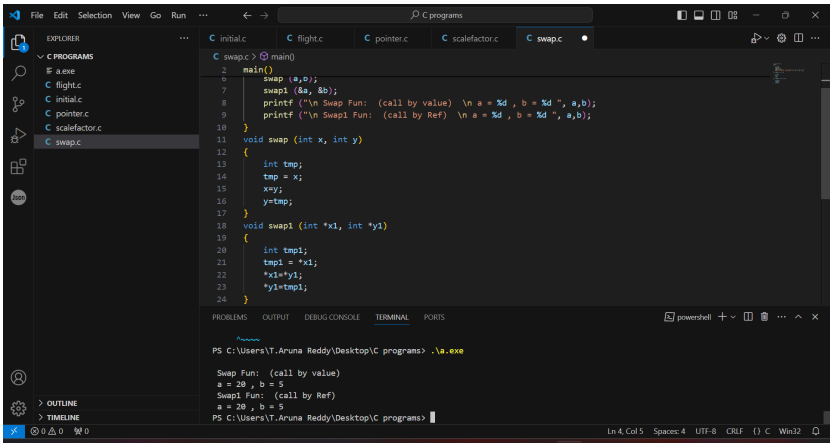
```c
#include <stdio.h>
#include <conio.h>

main()
```

```
{
   printf ("No. of Bytes occupied by int is %d \n", sizeof(int));
   printf ("No. of Bytes occupied by float is %d \n", sizeof(float));
   printf ("No. of Bytes occupied by double is %d \n", sizeof(double));
   printf ("No. of Bytes occupied by char is %d \n", sizeof(char));
   getch();

}
```



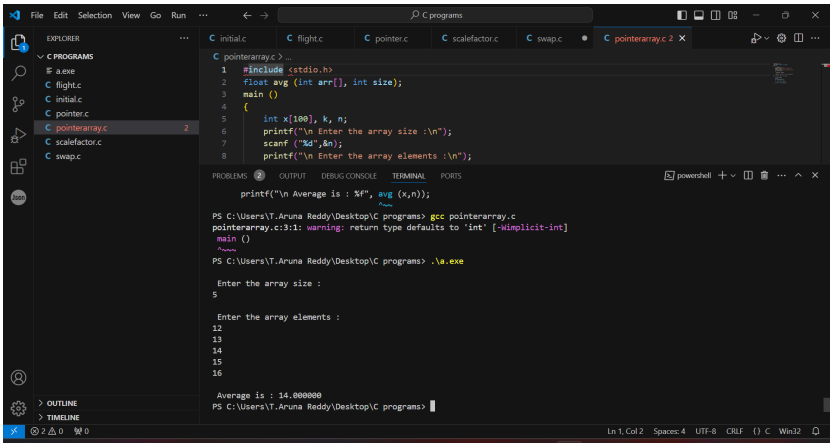
3.swap

```
#include <stdio.h>
main()
{
   int a,b;
   a=5, b=20;
   swap (a,b);
   swap1 (&a, &b);
   printf ("\n Swap Fun:  (call by value)  \n a = %d , b = %d ", a,b);
   printf ("\n Swap1 Fun:  (call by Ref)  \n a = %d , b = %d ", a,b);
}
void swap (int x, int y)
{
   int tmp;
   tmp = x;
   x=y;
   y=tmp;
}
void swap1 (int *x1, int *y1)
{
   int tmp1;
   tmp1 = *x1;
   *x1=*y1;
   *y1=tmp1;
}
```



4.array pointer

```
#include <stdio.h>
float avg (int arr[], int size);
main ()
{
   int x[100], k, n;
   printf("\n Enter the array size :\n");
   scanf ("%d",&n);
   printf("\n Enter the array elements :\n");
   for (k=0;k<n;k++)
   {
     scanf("%d", &x[k]);
    }
    printf("\n Average is : %f", avg (x,n));
}
float avg (int arr[], int size)
{
   int *p,i,sum=0;
   p=arr;
   for (i=0;i<size;i++)
   {
      sum = sum + *(p+i);
   }
   return ((float) sum/size);
}
```

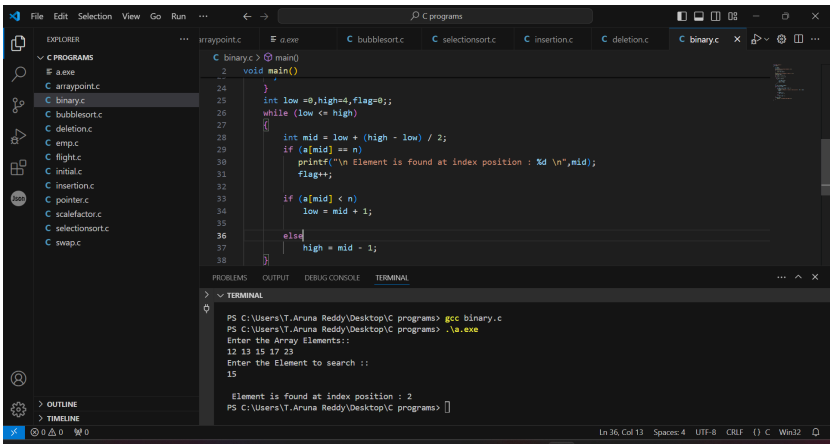


17th may

1.binary

```
#include <stdio.h>
void main()
{
   int a[5];
   int i,j,n;
   printf("Enter the Array Elements::\n");
   for (i=0;i<5;i++)
   {
      scanf("%d",&a[i]);
   }
   printf("Enter the Element to search ::\n");
   scanf("%d",&n);
   for (i=0; i < 5-1; i++)
   {
   for (j=0; j < 5-i-1; j++)
     {
     if (a[j] > a[j+1])
       {
           int temp = a[j];
           a[j] = a[j+1];
           a[j+1] = temp;
       }
     }
   }
   int low =0,high=4,flag=0;;
   while (low <= high)
   {
      int mid = low + (high - low) / 2;
      if (a[mid] == n)
         printf("\n Element is found at index position : %d \n",mid);
         flag++;
```

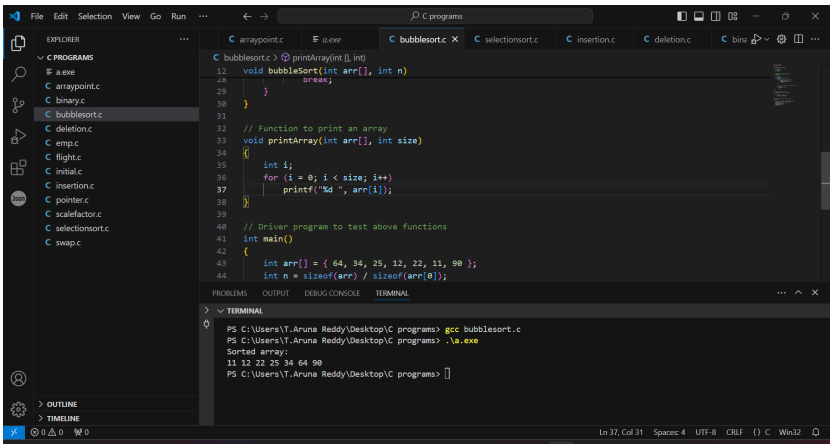
```
        if (a[mid] < n)
            low = mid + 1;

        else
            high = mid - 1;
    }
    if(flag ==0){
        printf(" \n Element NOT FOUND \n");
    }
}
```



2.bubble sort

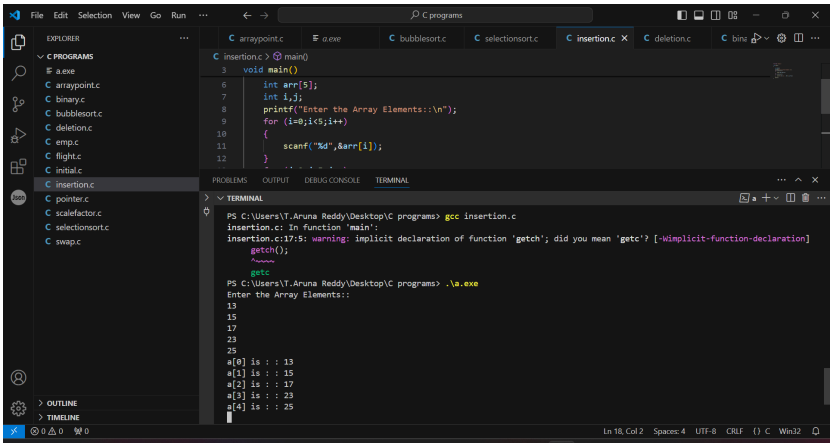
```
#include <stdbool.h>
#include <stdio.h>

void swap(int* xp, int* yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

// An optimized version of Bubble Sort
void bubbleSort(int arr[], int n)
{
    int i, j;
    bool swapped;
    for (i = 0; i < n - 1; i++) {
        swapped = false;
        for (j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                swap(&arr[j], &arr[j + 1]);
```

```
            swapped = true;
        }
    }

    // If no two elements were swapped by inner loop,
    // then break
    if (swapped == false)
        break;
  }
}

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

// Driver program to test above functions
int main()
{
  int arr[] = { 64, 34, 25, 12, 22, 11, 90 };
  int n = sizeof(arr) / sizeof(arr[0]);
  bubbleSort(arr, n);
  printf("Sorted array: \n");
  printArray(arr, n);
  return 0;
}
```



3.Insertion

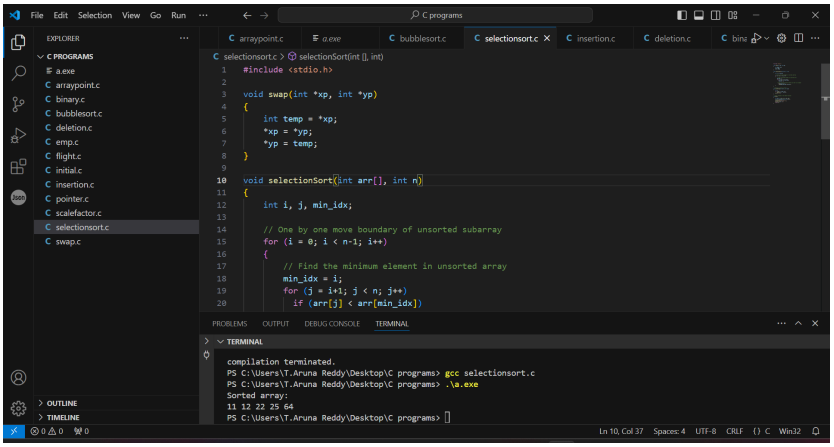
```
#include <stdio.h>

void main()
{

    int arr[5];
    int i,j;
    printf("Enter the Array Elements::\n");
    for (i=0;i<5;i++)
    {
        scanf("%d",&arr[i]);
    }
    for (j=0;j<5;j++)
    {
        printf("a[%d] is : : %d\n",j,arr[j]);
    }
    getch();
}
```



4.Selection sort

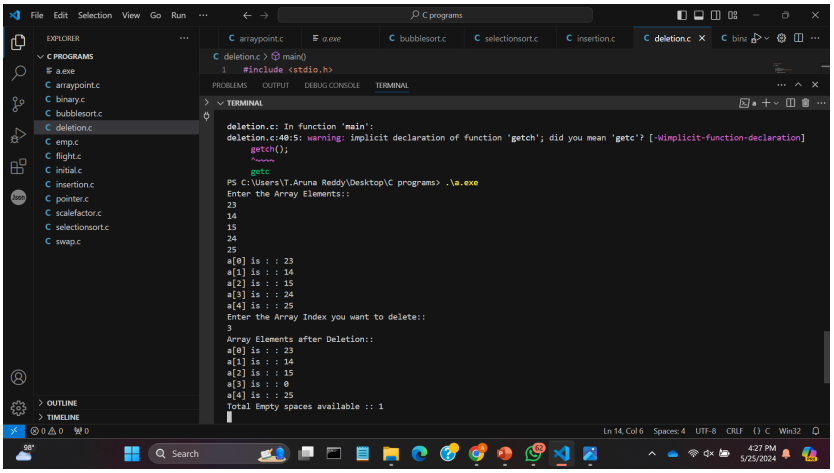
```
#include <stdio.h>

void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

void selectionSort(int arr[], int n)
{
    int i, j, min_idx;

    // One by one move boundary of unsorted subarray
```

```
    for (i = 0; i < n-1; i++)
    {
        // Find the minimum element in unsorted array
        min_idx = i;
        for (j = i+1; j < n; j++)
          if (arr[j] < arr[min_idx])
            min_idx = j;

        // Swap the found minimum element with the first element
           if(min_idx != i)
            swap(&arr[min_idx], &arr[i]);
    }
}

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

// Driver program to test above functions
int main()
{
    int arr[] = {64, 25, 12, 22, 11};
    int n = sizeof(arr)/sizeof(arr[0]);
    selectionSort(arr, n);
    printf("Sorted array: \n");
    printArray(arr, n);
    return 0;
}
```



5.Deletion

```
#include <stdio.h>

void main()
{

    int arr[5];
    int i,j,n,counter=0;
    printf("Enter the Array Elements::\n");
    for (i=0;i<5;i++)
    {
        scanf("%d",&arr[i]);
    }
    for (j=0;j<5;j++)
    {
        printf("a[%d] is : : %d\n",j,arr[j]);
    }

    printf("Enter the Array Index you want to delete::\n");
    scanf("%d",&n);
    arr[n] = 0;
    /*for (i=0;i<=n;i++)
    {
        if(i==n)
        {
            arr[i]= 0;
        }
    }*/
    printf("Array Elements after Deletion::\n");
    for (j=0;j<5;j++)
    {
        printf("a[%d] is : : %d\n",j,arr[j]);
    }

    for (i=0;i<5;i++)
    {
        if (arr[i]== 0)
            counter = counter +1;
    }
    printf("Total Empty spaces available :: %d\n", counter);
    getch();
}
```



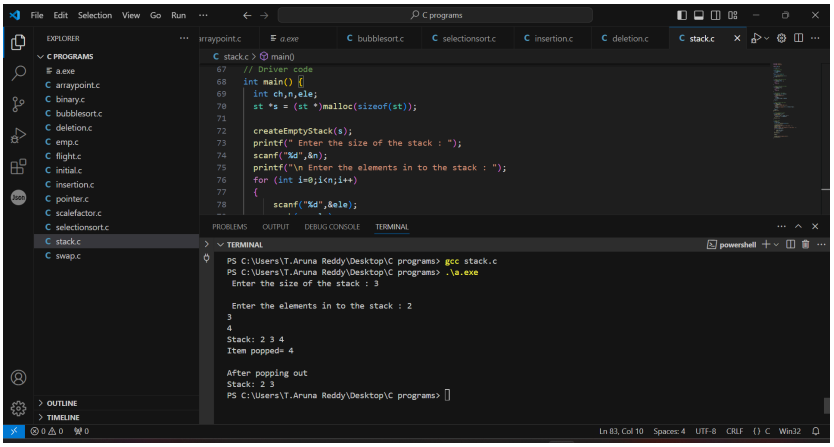
19-24 may

1.Stack implementation

```
#include <stdio.h>
#include <stdlib.h>

#define MAX 10

int count = 0;

// Creating a stack
struct stack {
  int items[MAX];
  int top;
};
typedef struct stack st;

void createEmptyStack(st *s) {
  s->top = -1;
}

// Check if the stack is full
int isfull(st *s) {
  if (s->top == MAX - 1)
    return 1;
  else
    return 0;
}

// Check if the stack is empty
int isempty(st *s) {
  if (s->top == -1)
    return 1;
  else
```

```
    return 0;
}

// Add elements into stack
void push(st *s, int newitem) {
  if (isfull(s)) {
    printf("STACK FULL");
  } else {
    s->top++;
    s->items[s->top] = newitem;
  }
  count++;
}

// Remove element from stack
void pop(st *s) {
  if (isempty(s)) {
    printf("\n STACK EMPTY \n");
  } else {
    printf("Item popped= %d", s->items[s->top]);
    s->top--;
  }
  count--;
printf("\n");
}

// Print elements of stack
void printStack(st *s) {
  printf("Stack: ");
  for (int i = 0; i < count; i++) {
    printf("%d ", s->items[i]);
  }
  printf("\n");
}

// Driver code
int main() {
  int ch,n,ele;
  st *s = (st *)malloc(sizeof(st));

  createEmptyStack(s);
  printf(" Enter the size of the stack : ");
  scanf("%d",&n);
  printf("\n Enter the elements in to the stack : ");
  for (int i=0;i<n;i++)
  {
      scanf("%d",&ele);
      push(s, ele);
```

```
  }
 printStack(s);

  pop(s);

  printf("\nAfter popping out\n");
  printStack(s);
}
```



2.Queue implementation

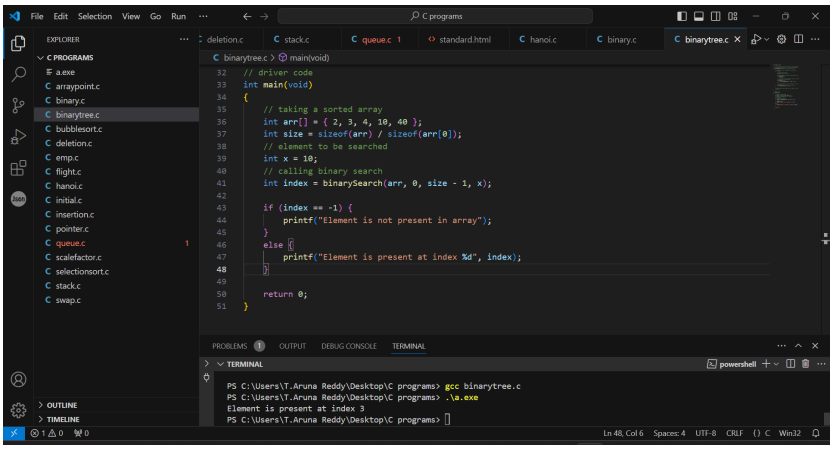
```
#include <stdio.h>
#define SIZE 5

void enQueue(int);
void deQueue();
void display();

int items[SIZE], front = -1, rear = -1;

int main() {
  //deQueue is not possible on empty queue
  deQueue();

  //enQueue 5 elements
  enQueue(1);
  enQueue(2);
  enQueue(3);
  enQueue(4);
  enQueue(5);

  // 6th element can't be added to because the queue is full
  enQueue(6);

  display();
```

```
  //deQueue removes element entered first i.e. 1
  deQueue();
 //Now we have just 4 elements
  display();

  return 0;
}

void enQueue(int value) {
  if (rear == SIZE - 1)
    printf("\nQueue is Full!!");
  else {
    if (front == -1)
      front = 0;
    rear++;
    items[rear] = value;
    printf("\nInserted -> %d", value);
  }
}

void deQueue() {
  if (front == -1)
    printf("\nQueue is Empty!!");
  else {
    printf("\nDeleted : %d", items[front]);
    front++;
    if (front > rear)
      front = rear = -1;
  }
}

// Function to print the queue
void display() {
  if (rear == -1)
    printf("\nQueue is Empty!!!");
  else {
    int i;
    printf("\nQueue elements are:\n");
    for (i = front; i <= rear; i++)
      printf("%d  ", items[i]);
  }
  printf("\n");
}
```

3.binary tree

```
#include <stdio.h>

// A recursive binary search function. It returns location
// of x in given array arr[l..r] if present, otherwise -1
int binarySearch(int arr[], int l, int r, int x)
{
    // checking if there are elements in the subarray
    if (r >= l) {

        // calculating mid point
        int mid = l + (r - l) / 2;

        // If the element is present at the middle itself
        if (arr[mid] == x)
            return mid;

        // If element is smaller than mid, then it can only
        // be present in left subarray
        if (arr[mid] > x) {
            return binarySearch(arr, l, mid - 1, x);
        }

        // Else the element can only be present in right
        // subarray
        return binarySearch(arr, mid + 1, r, x);
    }

    // We reach here when element is not present in array
    return -1;
}

// driver code
int main(void)
{
    // taking a sorted array
    int arr[] = { 2, 3, 4, 10, 40 };
    int size = sizeof(arr) / sizeof(arr[0]);
    // element to be searched
    int x = 10;
    // calling binary search
    int index = binarySearch(arr, 0, size - 1, x);

    if (index == -1) {
        printf("Element is not present in array");
    }
    else {
```

```
        printf("Element is present at index %d", index);
    }

    return 0;
}
```



4.tower of honai