1. **Activity: Design structure to represent complex numbers. Get two complex numbers and find sum of them using a function to which structure is passed as arguments.**

**Algorithm:**

1. Get two complex numbers input from user
2. Store the values in two variable structures
3. Pass them as argument to comp\_sum function
   1. Add real part to real part
   2. Add imaginary part to imaginary part
   3. Return the sum
4. Print out the sum.

**Test Case Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Complex number 1 | Complex number 2 | Sum printed out | Pass/Fail |
| 3 | 4 | 7 + 0i | Pass |
| 2 + 3i | 5 + 6i | 7 + 9i | Pass |
| 1 + 5i | 2 + i | 3 + 6i | Pass |

**Program:**

*#include* <stdio.h>

struct complex{

    int real;

    int comp;

};

struct complex comp\_sum(struct complex c1, struct complex c2){

    struct complex t;

    t.real = c1.real + c2.real;

    t.comp = c1.comp + c2.comp;

*return* t;

}

int main(){

    struct complex c1, c2, sum;

    printf("Enter complex no. 1\n");

    printf("Real Part: ");

    scanf("%d", &c1.real);

    printf("Complex Part: ");

    scanf("%d", &c1.comp);

    printf("\n");

    printf("Enter complex no. 2\n");

    printf("Real Part: ");

    scanf("%d", &c2.real);

    printf("Complex Part: ");

    scanf("%d", &c2.comp);

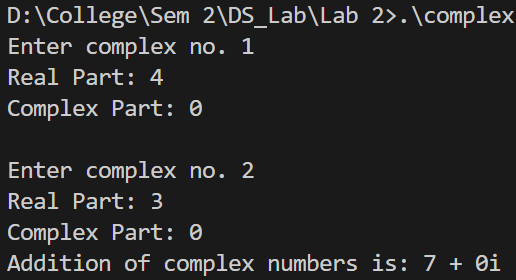
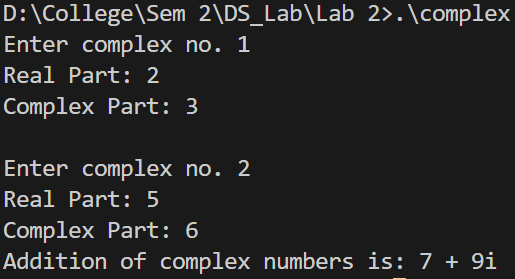
    sum = comp\_sum(c1, c2);

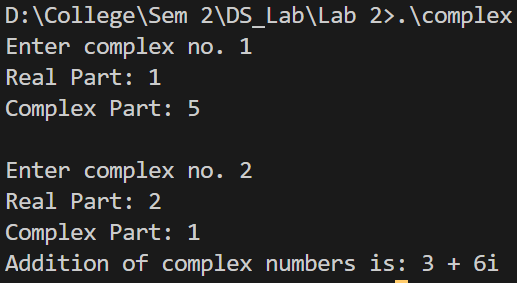
    printf("Addition of complex numbers is: %d + %di", sum.real, sum.comp);

*return* 0;

}

**Screenshot of compilation and execution:**

** **

****

1. **Activity:** Design a structure named “student\_record” …….. is dynamically allocated.

**Program:**

*#include* <stdio.h>

*#include* <stdlib.h>

struct date

{

    int dd;

    int mm;

    int yy;

};

struct student\_record

{

    char fname[20];

    char lname[20];

    int sap;

    int enroll;

    struct date dbirth;

    struct date dregister;

    int marks[5];

};

int main(){

    int n;

    printf("Enter number of students: ");

    scanf("%d", &n);

    struct student\_record \*ptr, \*curr;

*//Memory allocation*

    ptr = (struct student\_record \*)malloc(n\*sizeof(struct student\_record));

*//Getting input*

*for* (int i = 0; i < n; i++)

    {

        printf("Enter Student %d data ->\n", i+1);

        curr = (ptr+i);

*//Name*

        printf("Enter first name: ");

        scanf("%s", curr->fname);

        printf("Enter last name: ");

        scanf("%s", curr->lname);

*//SAP*

        printf("Enter SAP-ID: ");

        scanf("%d", &curr->sap);

*//Enrollment*

        printf("Enter enrollment number: R");

        scanf("%d", &curr->enroll);

*//Marks of 5 subjects*

        printf("Enter 5 marks: ");

        scanf("%d %d %d %d %d", &curr->marks[0], &curr->marks[1], &curr->marks[2], &curr->marks[3], &curr->marks[4]);

*//Dates*

        printf("\nNote: Enter date in DD MM YYYY format\n");

        printf("Enter date of birth: ");

        scanf("%d %d %d", &curr->dbirth.dd, &curr->dbirth.mm, &curr->dbirth.yy);

        printf("Enter date of registration: ");

        scanf("%d %d %d", &curr->dregister.dd, &curr->dregister.mm, &curr->dregister.yy);

    }

    printf("\n\n");

*//Printing Report cards*

*for* (int i = 0; i < n; i++)

    {

        curr = ptr + i;

        printf("|\t\tReport Card\t\t |\n");

        printf("| Name: %s %s\t| SAP-ID: %d\t |\n", curr->fname, curr->lname, curr->sap);

        printf("| Enrollment No: R%d\t\t\t |\n", curr->enroll);

        printf("| Date of Birth: %d-%d-%d\t\t\t |\n", curr->dbirth.dd, curr->dbirth.mm, curr->dbirth.yy);

        printf("| Date of Registration: %d-%d-%d\t\t |\n", curr->dregister.dd, curr->dregister.mm, curr->dregister.yy);

        printf("|\t-\t-\t-\t-\t |\n");

        printf("|Subject 1\t\t|\t\t%d|\n", curr->marks[0]);

        printf("|Subject 2\t\t|\t\t%d|\n", curr->marks[1]);

        printf("|Subject 3\t\t|\t\t%d|\n", curr->marks[2]);

        printf("|Subject 4\t\t|\t\t%d|\n", curr->marks[3]);

        printf("|Subject 5\t\t|\t\t%d|\n", curr->marks[4]);

        printf("|\t-\t-\t-\t-\t |\n");

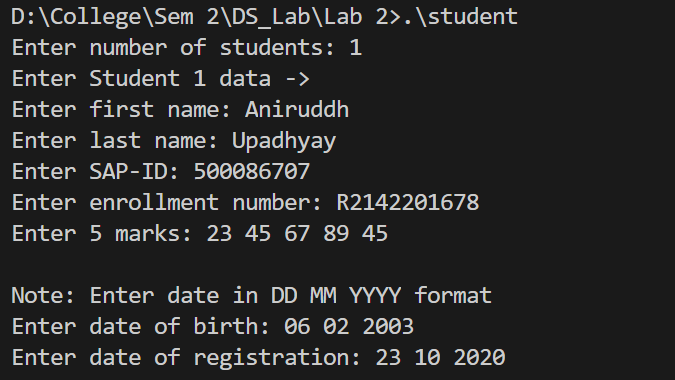
        printf("\n\n");

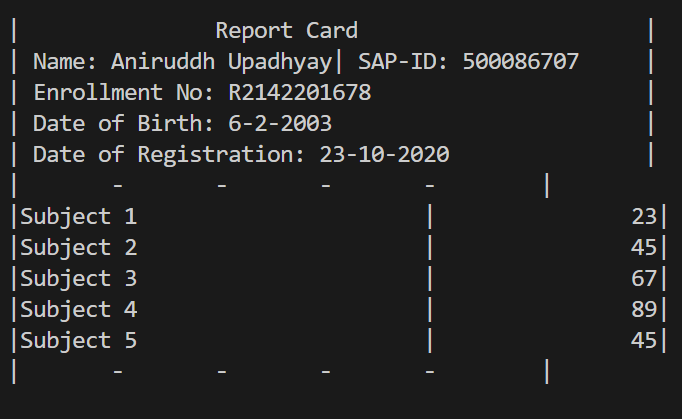
    }

*return* 0;

}

**Screenshot of compilation and execution:**





1. **Activity:** Consider 2 quadratic eqautions…….the quadratic equations Eq1 and Eq2.

**Algorithm:**

1. Get equation from user
2. Call the solve function that uses the quadratic formula to get the roots of the passed equation (user given).
3. After we get the 2 non-common roots, we put that into the equation for the sum of product of roots takin 2 at a time. i.e r1^2 + 2r1(r2+r4) + (r2 x r4) = sum.
4. Solve the formed equation by passing it into the solve function. Choose the largest possible root as the common root.
5. Make the equations using the relation between roots and coefficients.

**Program:**

*#include* <stdio.h>

*#include* <math.h>

struct eqn {

    float a, b, c, r1, r2;

};

void eq\_make(struct eqn \*p){

    p->a = 1;

    p->b = -(p->r1 + p->r2);

    p->c = (p->r1)\*(p->r2);

}

void solve(struct eqn \*p){

    int d = (p->b)\*(p->b) - 4 \* (p->a) \* (p->c);

*if*(d>0){

        p->r1 = (-(p->b) + sqrt(d))/2\*(p->a);

        p->r2 = (-(p->b) - sqrt(d))/2\*(p->a);

    }

}

int main(){

    struct eqn first, second, third, SOP, \*ptr;

    int sum;

    ptr = &third;

*//Input user-defined equation*

    printf("Enter a quadratic equation ->\n");

    printf("Coefficient of x^2: ");

    scanf("%f", &ptr->a);

    printf("Coefficient of x: ");

    scanf("%f", &ptr->b);

    printf("Constant: ");

    scanf("%f", &ptr->c);

*//Solve user defined equation and get two wanted roots.*

    solve(&third);

    printf("Roots are: %.2f, %.2f\n", third.r1, third.r2);

*//Let r2 = r5 and r4 = r6*

    first.r2 = third.r1;

    second.r2 = third.r2;

*//Getting required sum*

    printf("Enter sum of product of all roots taken two at a time: ");

    scanf("%d", &sum);

*//By derivation of SOP formula we have r1^2 + 2r1(r2+r4) + (r2\*r4) = sum*

    SOP.a = 1;

    SOP.b = 2\*((first.r2)+(second.r2));

    SOP.c = (first.r2)\*(second.r2) - sum;

    solve(&SOP);

    printf("Equation formed: %.2fx^2 + %.2fx + %.2f = 0\n", SOP.a, SOP.b, SOP.c);

    float common = (SOP.r1 > SOP.r2) ? SOP.r1 : SOP.r2;

    printf("Chosen common root is %f\n\n", common);

    first.r1 = common;

    second.r1 = common;

    eq\_make(&first);

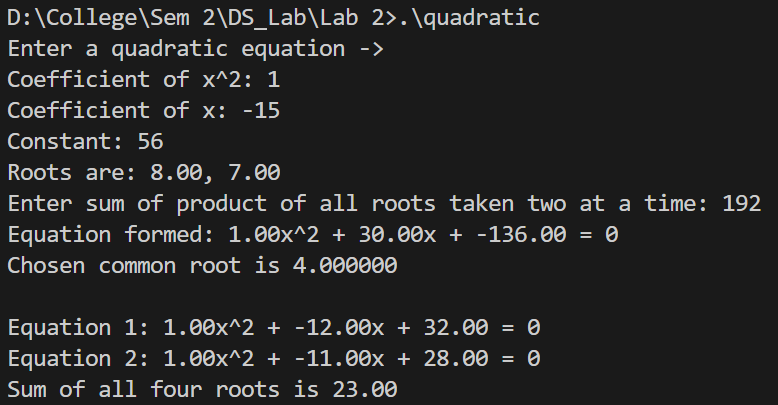
    eq\_make(&second);

    printf("Equation 1: %.2fx^2 + %.2fx + %.2f = 0\n", first.a, first.b, first.c);

    printf("Equation 2: %.2fx^2 + %.2fx + %.2f = 0\n", second.a, second.b, second.c);

    printf("Sum of all four roots is %.2f", first.r1 + first.r2 + second.r1 + second.r2);

}

**Screenshot of compilation and execution: **

1. **Activity:** Design a union ‘product’ …. Display all the details of the procured product

**Algorithm:**

1. Create a union and start taking user inputs.
2. Scan user inputs like Name, price per unit, number of quantities.
3. Store these values also in local variables to display at end.
4. Calculate total amount by Price x Quantity
5. Print all details

**Program:**

*#include* <stdio.h>

*#include* <string.h>

union product

{

    char name[50];

    float pricePerUnit;

    int noOfItems;

};

void main(){

    union product pro, \*b;

    b = &pro;

    char pname[50];

    printf("Enter product name: ");

    scanf("%s", b->name);

    strcpy(pname, b->name);

    printf("Enter price of one unit: ");

    scanf("%f", &pro.pricePerUnit);

    float x = b->pricePerUnit;

    printf("Enter number of item purchased: ");

    scanf("%d", &pro.noOfItems);

    int y = b->noOfItems;

    printf("\nTotal amount to be paid: %.2f\n\n", x\*y);

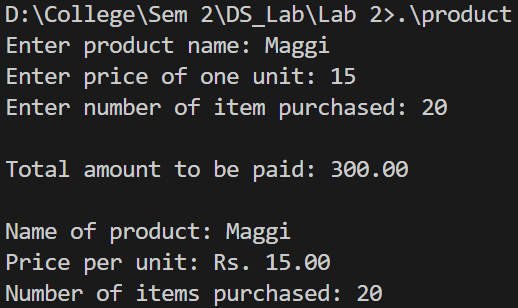
    printf("Name of product: %s\n", pname);

    printf("Price per unit: Rs. %.2f\n", x);

    printf("Number of items purchased: %d", y);

}

**Screenshot of compilation and execution:**

****