Hand Book
On
Appearing at Infosys
Aspiration 2020
Teaser Round Contest

Gandhi Institute For Technology
Bhubaneswar

Aspirations2020 Programming Contest - Rules

Prove yourself to be a Smart Programmer!
Here is the opportunity to demonstrate your programming skills.
Team up with your best friends and get ready for online Programming contest!

Infosys is pleased to announce **Aspirations2020** – **Programming Contest** for engineering and MCA students. This is organized by Campus Connect team of Infosys and sponsored by Infosys Technologies Limited.

Aspirations2020 is a platform for students of engineering colleges to prepare themselves to become smart professionals.This contest under Aspirations2020 will encourage the spirit of competitiveness and accelerate learning through extra curricular activities, within the student community.

Participate and enhance your programming competency
Programming Contest Resource :
http://www.devsquare.com/adc/dsweb/contest/campusconnect/index.jsp

Rules and Regulations – Programming Contest:

1. Participation for this event is open for second, third and final year engineering and second & third year MCA students. Students from any engineering discipline/ branch may participate .
2. This is a team event. Each team will comprise of three members from the same institution. Team composition cannot change once the college round is completed.
3. You need to register for the contest as a participant on the Campus Connect Portal by providing necessary personal data, before the registration closing date. You will find some tips and other learning resources to help you prepare for the contest.
4. Once you register for the event, indicate your acceptance that the organizers decision will be final and binding.
5. Prizes and certificates will be distributed for winning teams from inter-college contest.College level winners will receive certificates.

Contest Stages:

STAGE I: Teasers to participating colleges

- Before the college level contest start you will need to participate in a teaser round contest running for one week.
- The aim of this feature is to
    1. Promote the programming contest
    2. Familiarize you with how to use the online environment for the programming contest by practicing to solve problems
    3. Enable the college faculty to filter 30 students from each participating college
- A custom web page will be created on the Campus Connect portal that will be accessible to you. You may choose your preferred slot for practice.
- During the teaser round you may take up to 10 hours of programming practice, online. You will be required to randomly solve problems of simple, medium and complex types that carry different maximum marks. All solutions submitted will be evaluated and scored by special software.
- You will be provided with programming examples and several useful learning resources for preparing for programming contest
- You may choose to use any of the programming languages like Java, C, C++ and C#.
- Based on your performance in the teaser round, you will be shortlisted for the college level contest.

STAGE 2: College Level Contest

- The faculty from your college will select 30 best performing students based on the students' score in the teaser round. These students would be forming teams of 3 members each.
- Online contest platform will be used for conducting programming contest over the Internet.
- This round may take 1 to 4 days depending on number of participating colleges in State.
- The college level contest will be of 3 hours duration
- The contest will be conducted in the computer lab of the college with proctoring support by the non-computer science college faculty.
- You will be required to solve 3 problems within this time which of varying complexity.
- 2 best performing teams will be selected for participation in inter-college round by automated evaluation of submitted solutions.
- There will be manual validation of solutions in case of tie-scores.

STAGE 3: Inter-College programming contest

- The Inter-college round will be conducted for each logical region, consisting group of colleges.
- Inter-College rounds, short listing criteria is,
  - In a DC, if there is one zone, 6 best teams will be shortlisted from each zone.
  - If there are 2 zones, 3 best teams will be shortlisted from each zone.
  - If there are 3 zones, 2 best teams will be shortlisted from each zone.
  - If there are 5 zones, 1 best team will be shortlisted from each zone.
- Here again teams would code solution to 3 of the 5 problems randomly presented by system in Java, C, C++ or C#. The test would be for duration of 3 hours and it would be proctored by respective college faculty.

STAGE 4: DC Level Programming Contest

- The DC level contest will be conducted at Infosys DC online computer labs using the same contest platform over Internet.
- Same contest rules apply. DC level contest will result in selection of winner and runner-up teams.

STAGE 5: National Level Programming Contest

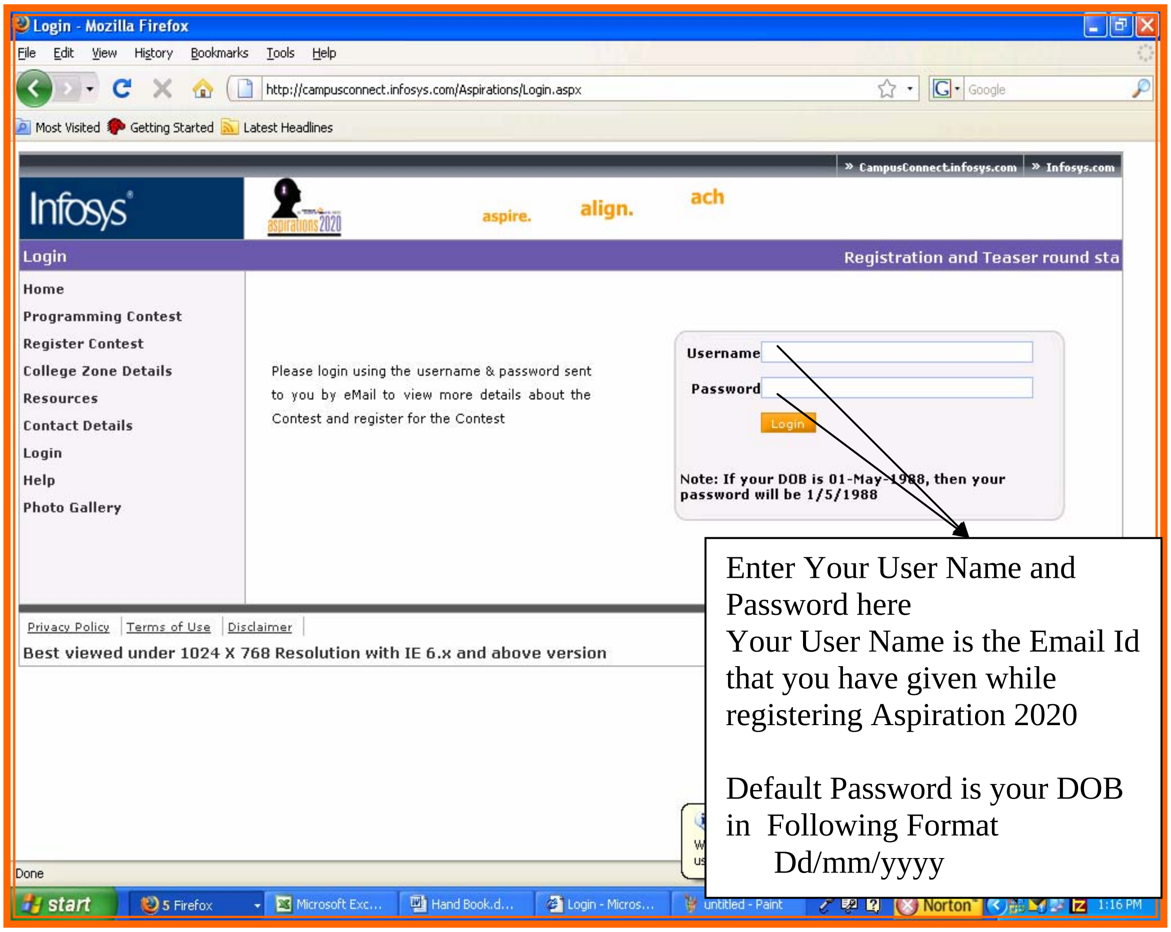
- The National level contest will be conducted at Infosys DC online computer labs using the same contest platform over Internet.
- Same contest rules apply. National level contest will result in selection of winner and runner-up teams.

How to start Teaser Round On-Line Test

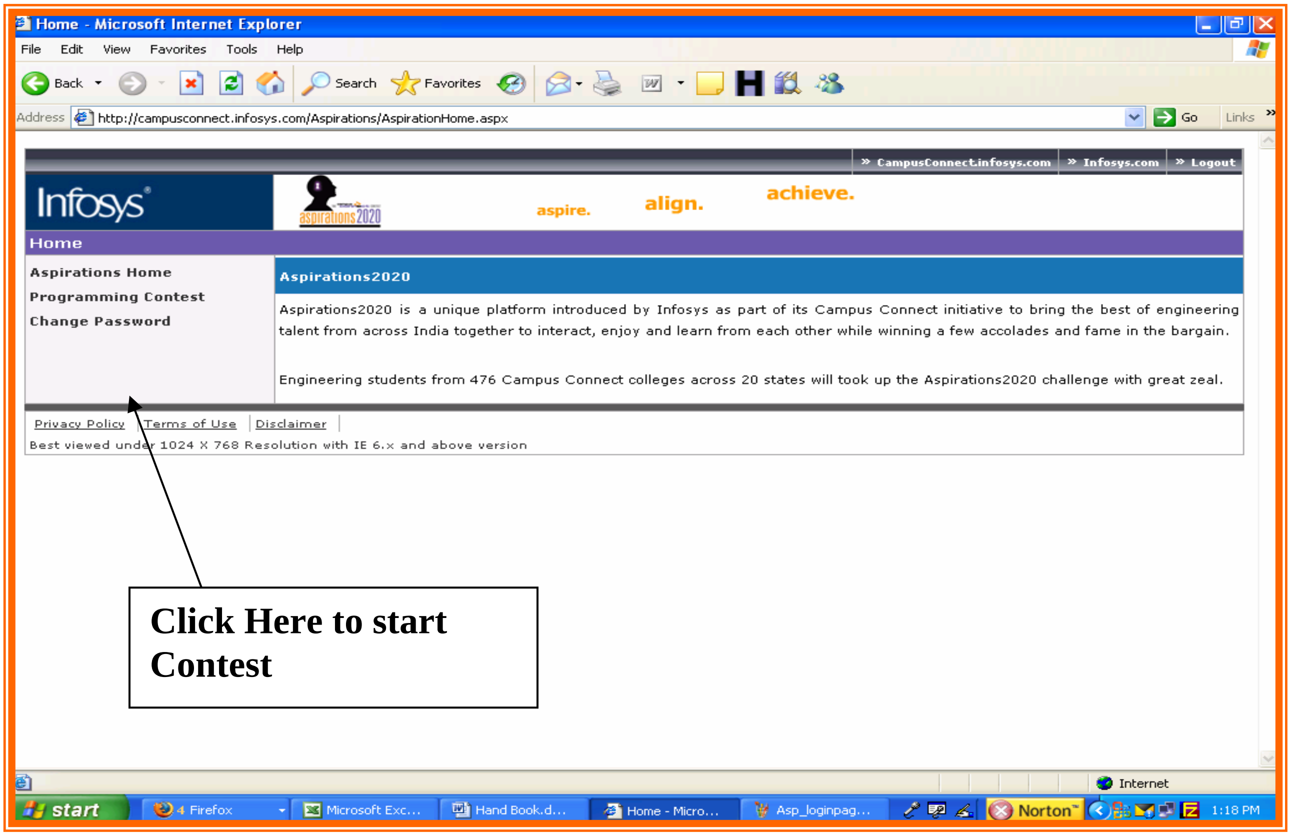
Step-1: First Login to you're a/c by giving your aspiration 2020 login user name and pass word.

Login Link Is:
http://campusconnect.infosys.com/Aspirations/Login.aspx

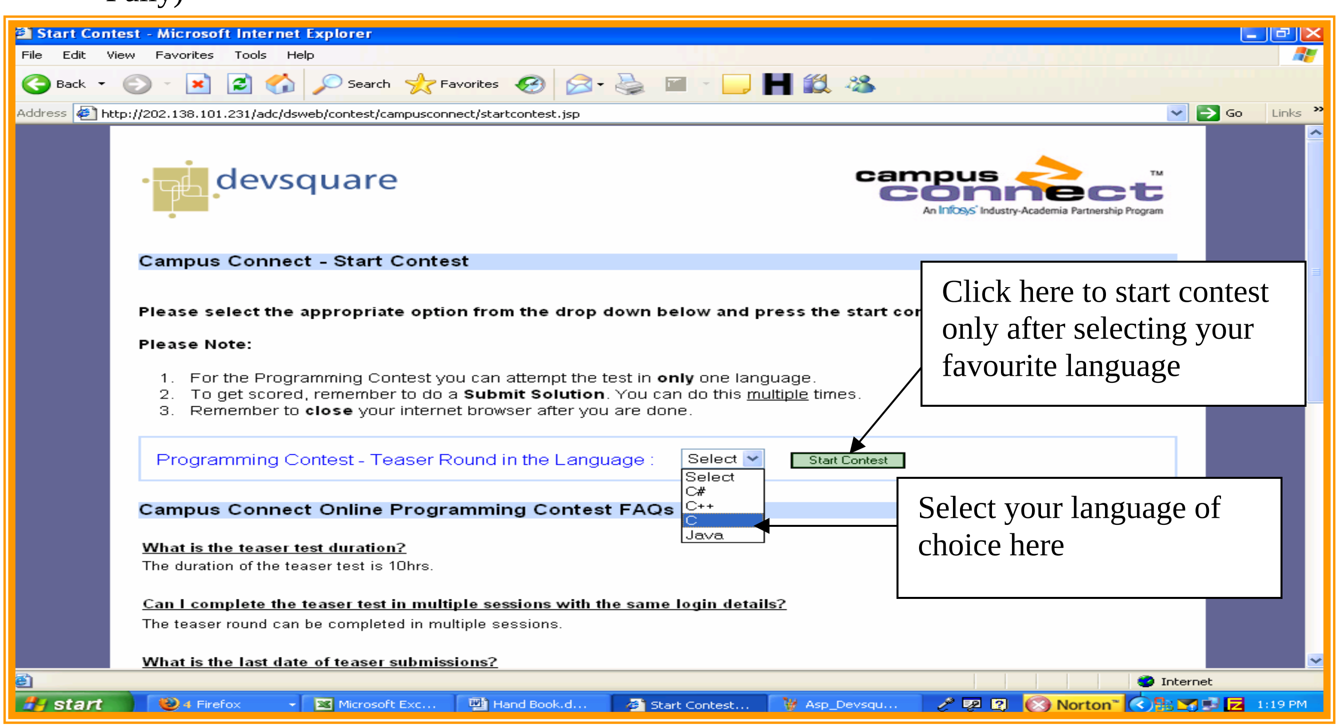
(Below is the View of your login Page)

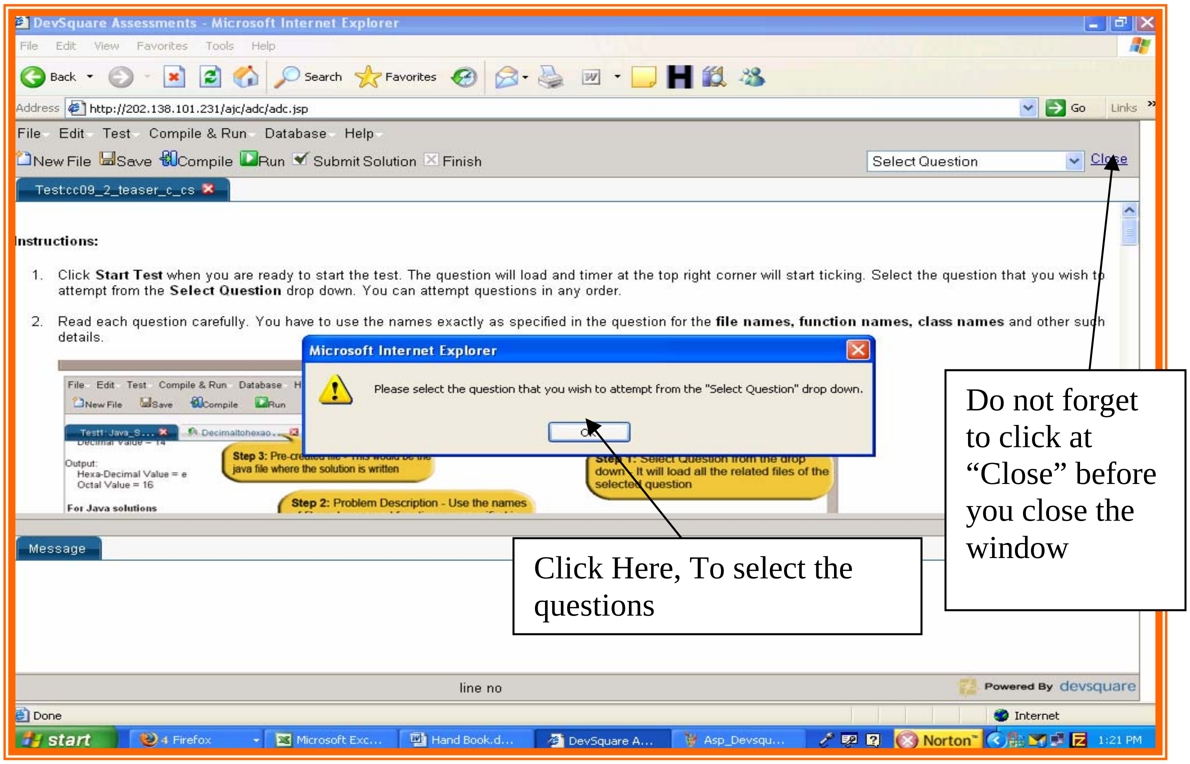


Step-2: You will directed to contest page click according to the following view

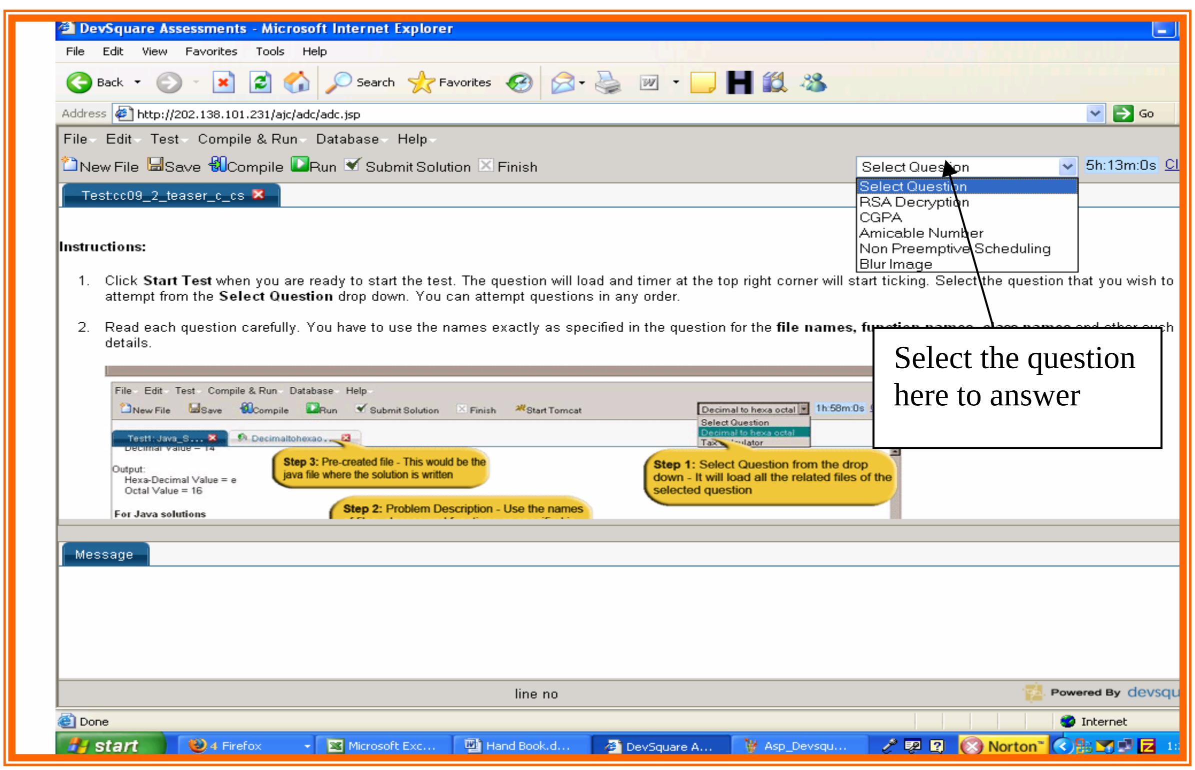


Step-3: Select your Programming Plat form and start contest(Read Instructions Care Fully)





Step-5: Read The instructions clearly and get the questions to prepare the program



Step-6: Read the question by click at the question in the question bar, then, the question will be displayed on the screen.(Your question is already given in this hand book)

Step-7: Prepare the program based on the problem statement. Design your program on pen and paper, then try it in normal complier.

Step-8: After Successful completion of the program, write the program on to the Devsquare Programming Development Platform, according to the instruction given there or ask for help to 9238313735(24hrs Open).

Step-9: You File name, variable names and function name must be as per the name suggested in the problem statement.

Step-10: after successful completion , save the file, then compile it, then submit it.

Do not click *Finish* Button, till you have completed and submitted all the program that you want to do.

Step -11: When you feel that, you finished and submitted all your answers, then click Finish button. Remember, once the finish button is clicked your test is completed.

For Any Assistance regarding problem statements or online examination, please feel free to contact the 24hrs **help line: 9238313735**

Following is the question Bank, from which Questions are randomly collected for your on line Test

Infosys Campus Connect Aspiration 2020 Programming Contest

Problem Statements

P1. RSA Decryption

RSA Encryption is the most used encryption method in the world right now. RSA is based on number theory concept called ***Modular Exponentiation***. For modular operation a number(n) is used which is multiplication of two(large) prime numbers P and Q. Here, we use the smallest valid prime number(p=11, q=3,n=33) for which the RSA algorithm works properly.

In this simple problem, you have to implement RSA decryption:

M= (c^d) mod n, where

C= cipherText, d=3, n=33

Please note that you can use the fact that

(a * b) mod n = ((a mod n) * (b mod n) to avoid calculating big numbers.

For example,

(10^5) mod 33 = (100 mod 33) * (100 mod 33) * (10 mod 33) = (1*1*10) mod 33 = 10

The function that you have to implement is

int getPlain(int cipherText)

Here the input to the function, cipherText, is the encrypted message represented as an integer and you have to return the decrypted value as per the above formula.

Constraints

The input to this function should be a zero or a positive number, else return -1.

Example-1

Input

cipherText=7

Output

Returns plaintext=13

Explanation

This output is resulted by decrypting the cipher text c=7 by using the following:

M= (c^d) mod n = 7^3 mod 33 = 343 mod 33 =13

Example-2

Input

cipherText=152

Output

Returns plaintext=14

Example-3

Input

cipherText=307

Output

Returns plaintext=10

For ‘C” Solutions

Header File: **rsadecrption.h**

Function Name: **int getPlain(int cipherText)**

File Name: **rsadecryption.c**

P2: CGPA

- Write a program to calculate the CGPA got by a student.
- You are given the SGPAs, credit for each semester and no.of semesters.
- CGPA, which is the cumulative grade point average is calculated on the basis of SGPA, which is the semester grade point average.
- CGPA is nothing but SUM of (SGPA * no.of credits in that semester)/SUM of all the credits.

The Prototype of the function is

- float calcCGPA(float sgpa[], int credits[], int noofsem)
- Where the sgpa is the float array contains the SGPAs of the student in different semesters, credits is the integer array contains the credits of the corresponding SGPA and the noOfSem is the number of semester. The function will return the CGPA.

Constraints

- All the sgpas are non negative, other wise return -1.
- All the credits are greater than zero, otherwise return -1
- The array length of sgpa and credits are equal to noOfSem.

Example -1

Input

sgpa={0.8,0.7,0.6,0.5} cedits={50,100,50,100}, noOfSem=4

Output

The function returns 0.633

Explanation

CGPA=(SUM(sgpa*credits)) / Sum(credits)

= (0.8 * 50 + 0.7 * 100 + 0.6 * 50 + 0.5 * 100)/300

Example-2

Input

Sgpa={7.2,8.32,7.5} credits={50,100,50}, noOfSem=3

Output

The function returns 7.835

Example-3

Input

Sgpa={7.2,8.32,7.5} credits={50,100,0} noOfSem=3

Output

The function returns -1

For ‘C” Solutions

Header File: **cgpa.h**

Function Name: **float clacCGPA(float sgpa[], int credits[], int noOfSem)**

File Name: **cgpa.c**

P3 Amicable Number

- Amicable numbers are two different number so related that the sum of the proper divisors of the one is equal to the other, one being considered as a proper divisor but not the number itself.
- Such a pair is(220,284); for the proper divisors of 220 are 1,2,4,5,10,11,20,22,44,55 and 110, of which the sum is 284; and the proper divisors of 284 are 1,2,4,71 and 142 of which the sum is 220.

The Prototype of the function

struct amicable *getAmicablePair(int Startnum, int endnum)

- Where Startnum represents starting integer number.
- Where endnum represents last integer number.
- The fnction returns a structure amicable where all the amicable pairs in between the startnum and endnum. Note that int ** in the structure will represent as 2 dimension array i.e; int[size][2].
- If the function not determine any pairs then returns NULL

Constraints

- The input startnum and endnum should be a positive integer else return NULL.
- The input startnum should be less than endnum else return NULL
- The input endnum should be less than 15000 else return NULL
- If the function not determine any pairs it should return NULL.

Example-1

Input

int startnum=10; int endnum=1000;

Output

getAmicablePairs returns {{220,284}}

Explanation

The proper divisors of 220 are 1,2,4,,5,10,11,20,22,44,55 and 110 of which the sum is 284; and the proper divisors of 284 are 1,2,4,71 and 142, of which the sum is 220.

Example-2

Input

int startnum=100; int endum=2000;

Output

getAmicablePairs( ) returns {{220,284},{1184,1210}}

Example-3

Input

int statnum=100; int endnum=10;

Output

getAmicablePairs( ) returns NULL

For 'C' Solutions

Header File: **amicablenumber.h**

Function Name: **struct amicable *getAmicablePair(int startnum, int endnum)**

File Name: **amicablenumber.c**

P 4: Non-Preemptive Scheduling

- Write a program to list the waiting time of the processes based on the arrival time and the burst time under non-premptive scheduling.
- In the non-premptive scheduling, the current process keeps the CPU unit it releases the CPU by terminating. The Operating system never intiates a context switch from a running processes to another process.
- Where there is more then one processes waiting for the CPU, the process which will release the CPU first by calculating the sum of arrival time and burst time, will be allocated.

The Prototype of the Function

int *getWaitingTime(int *process, int *arrivalTime, int *burstTime, int nprocess)

- The function getWaitingTime( ) takes the following as inputs, process which represents the number of processes.
- The function returns the waiting time of the process based on the arrival and burst time.

Constraints

- The arrival time of the first process should be start with 0, it never contain duplicate, the current process arrival time should be greater than prevous process else return NULL.
- The burst time should be generation 0 and less than 10 else return NULL.

Example-1

Input

int process[ ]= {1,2,3,4}

int arrivalTime[ ]={0,2,4,5}

int burstTime[ ] = {7,4,1,4}

int nProcess=4;

Output

The function getWaitingTime( ) returns {0,6,3,7}

Explanation

Process	**1**	**3**	**2**	**4**

0 7 8 12 16 start time of the process. The waiting time should be: 0,6,3,7

Example-2:

Input

int process[ ]={1,2,3};

int arrivalTime[ ]={0,4,8};

int burstTime[ ]={6,2,1};

Output

The function getWaitingTime( ) returns {0,2,0}

Example – 3

Input

int process[ ] = {1,2,3,4}

int arrivalTime[ ]={2,0,4,5}

int burstTime[ ]={7,4,1,4}

Output: The function getWaitingTime( ) returns NULL

For 'C' Solutions

Header Fle: **nonpreemptive.h**

Function Name: **int *getWaitingTime(int *process, int *arrivalTime, int *burstTime, int nprocess);**

File Name: **nonpreemptive.c**

P 5 Blur Image

- An image is often represented as a 2 dimensional array where each of the array index represents the color value at the corresponding pixels.
- The most popular format for the color is the RGB format where the color is represented as sum of three primary colors(Red, Green and Blue).
- Each of the R,G,B component can vary from 0-255(for the problem) and is represented in a single hexadecimal integer as 0xRRGGBB. Thus red is 0xFF0000 and Green is 0x00FF00 etc.,
- Blurring operation, used in image processing, involve calculating averages of areas of pixels in a source image for each of the final blurred image.

- In the blur image problem, you are given an image and you have to compute an image blurred as per the given parameters.
- If radius is r then for a pixel at data[x][y] in the output image, you would have to take an average of all pixels surrounding data[x][y] in radius r in the source image.
- For example if the radius 1, then data[3][3] in the output image would be average of the following q points in the source image(data[2][2],data[2][3], data[2][4],data[3][2],data[3][3], data[3][4], data[4][2],data[4][3],data[4][4]). Similarly for radius 2, you would taking average of upto 25points for each pixel(and for radius 3, average of up to 49 points).
- Note that averaging would be done individually for each R,G and B component i.e., each of R and G and B components should be separately averaged calculating the result pixel color.

The prototype of the function is:

struct image *blur_image(struct image *I, int radius)

- Where the function blur_image() takes a structure pointer of type image and an integer radius as input and returns the output image.
- Struct image consists of:

  data = the elements of 2D array

  rows = the no.of rows, and

  coloumn = no.of coloumns in the 2d array). In case of invalid input it returns NULL.

Constraints

- Each data point in the image is in RGB format(integer value up to 0xFFFFFF), if not return NULL.
- Radius r has to be less than both rows or column in the source image, otherwise return NULL.

Example-1

Input

```
               [6,12,18]
               [5,11,17]
Input ->data = [4,10,16]
               [3,9,15]
               [2,8,14]
```

Input -> rows = 5;

Input -> columns = 3;

Radius = 2;

Output

The function blur_image( ) returns

[11,11,11]

[10,10,10]

Input-> data = [10,10,10]

[9,9,9]

[9,9,9]

Input->rows = 5

Output -> columns=3

Example-2

Input

[0x 5a0060, 0x6a0050, 0x6a0050, 0x6a0050, 0x7f0038]

[0x 5a0060, 0x6a0050, 0x6a0050, 0x6a0050, 0x7f0038]

[0x 5a0060, 0x6a0050, 0x6a0050, 0x6a0050, 0x7f0038]

Input->rows = 3;

Input->columns=5;

Radius=1;

Output

The function blur_image( ) returns.

[0x 620058, 0x640055, 0x6a0050, 0x710048, 0x740044]

Output->data= [0x 620058, 0x640055, 0x6a0050, 0x710048, 0x740044]

[0x 620058, 0x640055, 0x6a0050, 0x710048, 0x740044]

Output->rows = 3

Output->columns=5

Example-3

Input

[0x5a0060, 0x6a0050, 0x6a0050, 0x6a0050, 0x7f0038]

Input->data= [0x5a0060, 0x6a0050, 0x6a0050, 0x6a0050, 0x7f0038]

[0x5a0060, 0x6a0050, 0x6a0050, 0x6a0050, 0x7f0038]

Input->rows = 3;

Input->column=5;

Radius=4;

Output

The function blur_image( ) returns NULL

C Solutoins

Header File: blurimage.h

Function Name: struct image *blur_image(struct image *i, int radius)

File Name: blurimage.c

P-6: Least Remaining Time

You have an internet browsing centre and because of some hardware problems only one machine is currently working fine. In order to control your customers to allow them to use that machine one by one for a given time slot. The person who has minimum balance browsing time is allowed to use the machine first and once the time slot or his browsing time is over then that person has to wait till every one has their turn.

All the users are now allowed in the minimum browsing left order. After the first turn is finished for all users, they will be allowed to use for their second turn with same constraint. This process wlll go on until all users have finished browsing. You have to return the schedule of the browsing in the order in which each person started to use the machine(For all their turns)

The Prototype of the Function is:

struct IntArray getSchedule(int browsingTime [ ], int noOfPersons, int timeSlot)

- Where browsingTime is an array of time the customer wants to browse, noOfPerson represents the size of the browsing time array and timeSlot is intinger value of time for each customer to access the machine.
- The Function returns structure IntArray, which contains the integer array of schedule(the order in which each person uses the machine) and the size of the array).

Constrints

- You can take the customer names as 1,2,3 and so on.
- The balance browsing time of each customer will be given in an array as same as order of customer name
- If there is any customer with lesser balance browsing period, then the common time slot, then he can browse until the common time slot expires.

Example-1

Input:

Balance_browsing_time = {10,7,3,4,11}
Number_of_persons = 5
Time_slot = 5

Output:

getSchedule( ) returns struct IntArray = (schedule = {3,4,2,1,5,2,1,5,5} and nSize=9}

Explanation

First arrange the customer based on balance browsing time. Thus the first user will be the third customer who has the balance browsing time of only 3. Then the next customer is fourth one who has the balance time of 4.

Then, second customer has the balance time 7 so the first time we will use the machine upto the time slot expires, then will wait in the queue. The last user is the 5th user having the execution time of 11 and he will use the machine upto the first time slot expires and goes to the queue. Now, all the users have used the machine once and they can again use it in the order of short remaining balance browsing. Here, second user has the shortest remaining time of 2 and he will using the machine and this process will continue up to all users despathched from the queue.

Example-2

Input:

Balance_browsing_time = {3,7,14,4,11}
Number_of_persons = 5

Time_slot = 7

Output

getSchedule( ) returns struct IntArray = (schedule = {1,4,2,5,3,5,3} and nSize=7}

For ‘C’ Solutions

Header File: leastremainingtime.h

Function Name: struct IntArray getSchedule(int browsingTime [ ], int noOfPersons, int timeSlot)

File Name: leastremainingtime.c

P-7: Black Jack

Black Jack is a very well known gambling card played against a dealer in a casino. In this card game, each player is trying to beat the dealer by obtaining a sum of card values not more than 21 which is greater than the sum of dealer card values. Player is initially given 2 cards, but he could choose to hit(ask for 3^{rd} card) or stand(no more cards). If he chooses to hit for 3^{rd} card and total score crosses 21 he get busted(loses irrespective of total score of dealer cards) face cards(Jack, Queens, and Kings) are worth 10 points. Aces are worth 1 or 11, which is preferable. Other cards are represented by their number.

Here, you have to implement a conservative strategy of playing black jack. This conservative player does not want to get busted and hit only when it’s safe to do so. He follows following strategy.
Hit if(score <=11) or (an ACE is held) stand otherwise.
Write a program to implement the above strategy given the initial 2 cards of the player.
The prototype of the function is:

int testHit(char firstcard, char secondcard)

Where the function testHit takes 2 character input Firstcard, Secondcard and return integer output.
Constraints

- Color of the card does not affect the score of the card.
- If the value of card is not valid then return -1.
- Does not consider the card with value 10.
- Input for face card with value 10.
- Input for face cards must be capital letters(‘A’,’Q’,’J’,’K’) else return -1.

Input Specifications

In input you will be given 2 single character representing the two cards.

Output Specifications

The output will be 1 or 0.

1 means the player can hit and 0 means stand.

Example -1:

Input

Firstcard = ‘J’, Secondcard=’5’

Output

The function returns 0

Explanation

Here the value of J(Jack) is 10. so the sum of card values is 15, so he chooses to stand.

Example-2

Input

Firstcard = 4 secondcard = 3

Output

The function return 1

Explanation

Here is the sum of values of card 4 and 3 is 7. It is less than 11, so it makes sense for him to draw another card.

Exmple-3:

Input

Firstcard = ‘A’ secondcard=’F’

Output

The function returns -1

Explanation

Here, F is an invalid card name

For ‘C’ Solutions

Header File: blackjack.h
Function Name: int testHit(char firstcard, char secondcard)
File Name: blackjack.c

P-8 Cell Phone Usage

One cell phon service provider launched following plan for SMS. For every user, 100 SMS are free for each recharged talk time.
Write a program to calculate the maximum friends a user can contact for the given talk time, SMS cost and call cost.

The Prototype of the function is

int numberOfContacts(int smsCost, int callCost,int talktime)

Where smsCost and callcost are the cost of SMS and a call in rupees respectively. The parameter talktime is the recharged value in rupees. The method will return the maximum number of friends that a user can use for the given talktime.

Constriants

- The talktime should be greater than Zero, otherwise return -1.
- The SMS cost should be greater than Zero, otherwise return -1
- The call cost should be greater than Zero, otherwise return -1.

Example-1

Input

smsCost=1, callCost=2, talktime =120

Output

The numberOfContacts( ) returns 220

Explanation

For the given talktime, number of SMS possible =120 /1 =120 and number of calls possible = 120/2 = 60. Hence, the maximum number of friends that an user can contact through this plan =100 free + 120 =220.

Example -2

Input

smsCost = 4, callCost = 2, talktime=120

Output

The function numberOfContacts( ) returns 160

Example -3

Input

smsCost = 4, callCost=2,talktime = 0

Output

The numberOfContacts( ) returns -1

For ‘C’ Solutions

Header File: cellphoneusage.h

Function Name: int numberOfContacts(int smsCost, int callCost,int talktime)
File Name: cellphoneusage.c

P-9: RSA Encryption

RSA Encryption is the MOST USED encryption method, in the world right now. RSA is based on number theory concept called modular exponentiation. For the modular operation a number(n) is used which is multiplication of two(large) prime numbers p and q.
Hence, we used the smallest valid prime number(p=11, q=3,n=33) for which RSA algorithm works properly.

In this simple problem, you have to implement the RSA encryption: C=(m^e) mod n, where m=plain text, e=7,n=33.

Please note that you can use the fact that(a*b) mod n = ((a mod n) * (b mod n)) mod n, to avoid calculating big numbers, for example, (10^5) mod 33 = ((100 mod 33) * (100 mod 33) * (100 mod 33)) mod 33 = (1 * 1* 10) mod 33 = 10.

The prototype of the function is:

int getCipher(int plaintext)

- Where plainText is the original message(m) represent as an integer.
- The function returns the encrypted form of m as per the above formulae.

Constraints

- The input to this function should be zero or a positive number, else return -1.

Example-1

Input

Int plainText =13

Output

The function getCipher( ) returns 7.

Explanation

This output is resulted by encrypting the message m=13, by using following: C= (m^e) mod n = (13^7) mod 33 = ((2197 mod 33) * (2197 mod 33) * (13 mod 33)) mod 33 = 7.

Example -2

Input

Int plainText =14

Output

The function getCipher( ) returns 20

Example -3

Input

Int plainText =27

Output

The function getCipher( ) returns 3

Example -4

Input

Int plainText =-91

Output

The function getCipher( ) returns -1

For ‘C’ Solutions:

Header File Name: rsaencryption.h

Function Name: int getCipher(int plainText)
File Name: rsaencryption.c

P-10: Chocolate Chips

Making chocolate chips cookies involves putting chocolate chips on a cookies dough, which is rolled into a plain about 50cm square. Circles of diameter 5cm are cut from the plain to _ from a chocolate chip cookies. Each chip is visible in the planar dough and you have to find the cookies which has the maximum number of chocolate chips inside the cookie(that is circle of 5cm)

The method chocolatechip contains 3 parameters. First one is X co-ordinate and second one is Y co-ordinates. which is of double array data type. All co-ordinates are in between 0.0(cm) and 50.0(cm). Each chip may be considered a small point(that is ignore the size of the chocolate chips). Each chip is at a different position. Third one is the number of co-ordinates. There are at most 200 chocolate chips(that is size of the first two arrays is less than 200).

Write a program to find the maximum number of chocolate chips that is contained in a single cookie of diameter 5 cm. The cookies have to be caught in the daugh, such that they are arranged symmetrically. (For Example, first 4 cookies have to be at centred at following co-ordinates: (2.5,2.5),(7.5,2.5),(2.5,7.5) and (7.5,7.5).

The prototype of the function is:

int ChocolateChips(double X[], double Y[], int noOfCoordinate)

-Where X is a double array of X co-ordinate, Y is double array of Y co-ordinate and noOfCoordintes represents the number of co-ordinates.

- The function returns integer value of the maximum number of chocolate chip present in a cookie.

Constraints:

- All co-ordinates are in between 0.0(cm) and 50.0(cm) else return -1.
- The no of coordinates should be greater than zero, else return -1.
- If there is no chocolate chip contained in any cookie of diameter 5cm, then return 0.
- Each chip may be consider a point. Each chip is at a different position.

Example 1:
Input:

int noofcoordinates=5(chocolate chips coordinates)

X[0]=1.0	Y[0]=1.0
X[1]=2.3	Y[1]=2.1
X[2]=2.8	Y[2]=3.0
X[3]=5.5	Y[3]=6.1
X[4]=7.0	Y[4]=8.9

Output

The function is return 3.

Explanation

First cookie which is in centred at (2.5,2.5) has 3 chocolate chips and second cookie which is in centered at (7.5,7.5) has 2 chocolate chips. Therefore 3 chocolate chips are the maximum.

Example -2:

Input

int noofcoordinates = 10

X[0]=1.0	Y[0]=1.0
X[1]=3.2	Y[1]=2.0
X[2]=4.5	Y[2]=3.0
X[3]=11.0	Y[3]=2.0
X[4]=1.4	Y[4]=2.5
X[5]=4.4	Y[5]=4.5
X[6]=2.8	Y[6]=3.5
X[7]=1.8	Y[7]=2.5
X[8]=13.4	Y[8]=4.5
X[9]=14.4	Y[9]=3.5

Output

The function returns 6

Explanation:

First cookie which is in centered at (2.5,2.5), has 6 chocolate chip {(1.0,1.0),(3.2,2.0),(4.5,3.0),(1.4,2.5),(2.8,3.5),(1.8,2.5)}and other cookie which is centered at (12.5,2.5) has 3 chocolate chip {(11.0,2.0),(13.4,4.5),(14.4,3.5)}and the chips(4.4,4.5) is outside of any cookie. Therefore 6 chocolate chips are the maximum.

For ‘C’ Solutions

Header File: Chocolate.h
Function Name: int Chocolatechips(double X[], double Y[], int noOfCoordinate)
File Name: Chocolate.c

P-11.Practical number:

A practical number is a (+ve) integer n such that all smaller (+ve) integer than n can be represent as sum of distinct divisors of n,ex-12 is a practical number because all the number from 1 to 11 can be expressed as sums of its divisiors 1,2,3,4,6:as well as these divisior themselves,we have 5=3+2,7=6+1,8=6+2,9=6+3,10=6+3+1,11=6+3+2.Any power to two is always a practical number & with the exception of 1 all other practical number are even.

In this problem,you have to find the practical numbers from a given range of integers.

The prototype of the function(s) is:

- int getNoOfPracticalNumbers(int from,int to)takes two integers & returns the number of practical numbers between the two input integers.
- Int* getPracticalNumbers(int from,int to)takes two integers & returns a pointer to an integer array which contains the practical numbers between the two input integers.

Constraints:

- The input values should be a non-zero (+ve) integers & the range can’t exceed 10000.
- For invalid input,”getNoOfPracticalNumbers”returns -1 & “getPracticalNumbers”returns NULL.

- If number of practical numbers are zero for the given range of input ,function “getPracticalNumbers”should return NULL.

Example-1:

Input

int from=1

int to=20

output

Function getNoOfPracticalNumbers returns:9

Function getPracticalNumbers returns:{1,2,4,6,8,12,16,18,20}

Explanation

The range of number is 1 to 20 & the method getNoOfPracticalNumbers return 9,i.e there are 9 numbers within satishfy the criteria for practical number.The method getPracticalNumbers returns all the numbers within that range which satishfy the criteria for practical number & they are {1,2,4,6,8,12,16,18,20},consider integer 12 & the divisiors for this number are 1,2,3,4,6,12.This number is a practical number because digits ranging from 1 to 11 can be derived by adding the divisors(we already have 1 & 2 as divisor & the remaining digits can be obtained by adding the follows 3=1+2,4=3+!,5=3+2,7=6+1,8=6+2,9=6+3,10=6+3+1,11=6+3+2)

Example-2

Input

int from=8000

int to=8200

output:-

Function getNoOfPracticalNumbers returns:27

FunctiongetPracticalNumbersreturns:{8000,8004,8008,8010,8034,8036,8040,8052,8060,8064,8080,8092,8096,8100,8112,8118,8120,8128,8136,8140,8148,8160,8176,8184,8190,8192,8200}

For C solution:

Header file:PracticalNumber.h

Function name: int getNoOfPracticalNumbers(int from,int to)

File name: PracticalNumbers.c

Best of your Hard Work