AMCAT Automata Questions with Answers

by FACE Prep · 8 days ago · 518 views

Aspiring Minds conducts their famous Automata Test to evaluate the candidates' programming skills beyond the correctness of the code or their ability to produce outputs. This test evaluates the candidates' knowledge of algorithms, data structures, recursion concepts, dynamic memory allocation and modular programming. Hence, most of the IT companies assess their candidates based on this test's performance. As this test plays a major role in the recruitment process, this article will discuss the AMCAT Automata Questions with answers, Tips and Strategies.

AMCAT Automata Questions

The AMCAT Automata test consists of 2 questions. One of these questions will be easy and the other will be hard to solve. The candidates should write the program in the editor, compile it and run test cases in the environment of Automata 2.0 software provided. The candidate has the freedom to choose the programming language to code in as the compiler supports wide range of programming languages.

Evaluation of Codes

The code written by the candidate is evaluated based on the following:

Whether the code runs for basic, advanced, and boundary test cases.
Whether the code industry-compatible.
Complexity of the code.
Warnings, run time and time out errors generated during the compilation.
Time taken to submit the final code.

AMCAT Automata Questions

The most commonly asked AMCAT Automata questions with their answers are discussed here. These questions of easy and hard difficulty levels. The commonly asked easy AMCAT Automata Questions are discussed below:

Question 1: Program to check if two given matrices are identical

Open in App (https://play.google.com/store/apps/details?id=in.faceprep.app&hl=en)

In C Language

```
#include <stdio.h>
#define N 4
// This function returns 1 if A[][] and B[][] are identical. Otherwise, it returns 0
int areSame(int A[][N], int B[][N])
{
int i, j;
for (i = 0; i < N; i++)
for (j = 0; j < N; j++)
if (A[i][j] != B[i][j]
return 0;
return 1;
}
int main()
{
int A[N][N] = { {1, 1, 1, 1},
{2, 2, 2, 2},
{3, 3, 3, 3},
{4, 4, 4, 4}};
int B[N][N] = { {1, 1, 1, 1},
{2, 2, 2, 2},
{3, 3, 3, 3},
{4, 4, 4, 4}};
if (areSame(A, B))
printf("Matrices are identical");
else
printf("Matrices are not identical");
return 0;
}
```

The above is the program in C language.

In Java

The same **program written in Java** is given below:

```
package org.sb.twoMatrices;
import java.util.Scanner;
public class TwoMatricesAreEqualOrNot {
private static int size;
public static int checkMatrices(int A[][], int B[][]) {
int i = 0, j = 0;
for (i = 0; i <= size - 1; i++) {
for (j = 0; j <= size - 1; j++) {
if (A[i][j] != B[i][j])
return 0;
}
}
return 1;
}
public static void main(String[] args) {
Scanner sc = null;
int i = 0, j = 0;
int result = 0;
// create scanner object
sc = new Scanner(in);
System.out.println("Please enter the size of array::");
size = sc.nextInt();
sc.nextLine();
int A[][] = new int[size][size];
int B[][] = new int[size][size];
for (i = 0; i <= size - 1; i++) {
for (j = 0; j <= size - 1; j++) {
System.out.println("Please enter Array A[" + i + "][" + j + "] value::");
A[i][j] = sc.nextInt();
sc.nextLine();
System.out.println("Please enter Array B[" + i + "][" + j + "] value::");
B[i][j] = sc.nextInt();
sc.nextLine();
} // inner for
} // outer for
// invoke the checkMatrices method
result = checkMatrices(A, B);
if (result == 1)
System.out.println("Matrices are identical..");
else
System.out.println("Matrices are not identical..");
// close scanner
sc.close();
}// main
}// class
```

Question 2: Program to print the given 2D Array or Matrix in spiral form

Input:

```
1   2   3   4   5   6
7   8   9   10  11  12
13  14  15  16  17  18
```

In C

```
#include <stdio.h>
#define R 3
#define C 6
```

Open in App (https://play.google.com/store/apps/details?id=in.faceprep.app&hl=en)

```
void spiralPrint(int m, int n, int a[R][C])
{
int i, k = 0, l = 0;
/* k - starting row index
m - ending row index
l - starting column index
n - ending column index
i - iterator */
while (k < m && l < n)
{
/* Print the first row from the remaining rows */
for (i = l; i < n; ++i)
{
printf("%d ", a[k][i]);
}
k++;
/* Print the last column from the remaining columns */
for (i = k; i < m; ++i)
{
printf("%d ", a[i][n-1]);
}
n--;
/* Print the last row from the remaining rows */
if ( k < m)
{
for (i = n-1; i >= l; --i)
{
printf("%d ", a[m-1][i]);
}
m--;
}
/* Print the first column from the remaining columns */
if (l < n)
{
for (i = m-1; i >= k; --i)
{
printf("%d ", a[i][l]);
}
l++;
}
}
}
/* Driver program for the above functions */
int main()
{
int a[R][C] = { {1, 2, 3, 4, 5, 6},
{7, 8, 9, 10, 11, 12},
{13, 14, 15, 16, 17, 18}
```

Open in App (https://play.google.com/store/apps/details?id=in.faceprep.app&hl=en)

```
};
spiralPrint(R, C, a);
return 0;
```

Output:

```
1 2 3 4 5 6 12 18 17 16 15 14 13 7 8 9 10 11
```

Question 3: A n-by-n matrix of 0’s and 1’s is given, where all 1’s in each row come before all 0’s, find the most efficient way to return the row with the maximum number of 0’s.

In C

```
#include <stdio.h>
#include <stdlib.h>
#include<conio.h>
#define COL4
#define ROW4 using namespace std;
intmain()
{int arr[ROW][COL]= { {1,1,1,1}, {1,1,0,0}, {1,0,0,0}, {1,1,0,0}, };
intrownum;
inti = 0, j = COL-1;
while(i<ROW &&j>;0)
{
if(arr[i][j]==0)
{
j--;
rownum=i;
}
elsei++;
}
printf("%d",rownum);
getch();
return 0;
}
```

Using Temp

```
#include <stdio.h>
#include <stdlib.h>
#define COL 4
#define ROW 4
int main() {
int arr[ROW][COL] = {
{1, 1, 1, 1},
{1, 1, 0, 0},
{1, 0, 0, 0},
{1, 1, 0, 0},
};
int rownum;
int i = 0, j = 0, temp, count = 0;
temp = count;
for (i = 0; i < 4; i++) {
count = 0;
for (j = 0; j < 4; j++) {
if (arr[i][j] == 0) {
count++;
}
if (count > temp) {
temp = count;
rownum = i;
}
}
}
printf("%d", rownum);
return 0;
}
```

Question 4: A Pythagorean triplet is a set of three integers a, b and c such that $a^2 + b^2 = c^2$. Given a limit, generate all Pythagorean Triples with values smaller than given limit. Limit = 20.

First, understanding the question is important. In this question, The output required is

```
3    4    5
8    6    10
5    12   13
15   8    17
12   16   20
```

This is because, $3^2 + 4^2 = 5^2$.

There are two ways to solve this question.

1) The simple way is to generate triplets smaller than t given limit and check if it meets the given condition every time. If the condition is true, print the triplets. Even though this method is simple, this is not effective as the code will be long for larger limits.

2) The effective way to solve is problem is by using the variables m and n instead of a, b and c by using the following relation:

$a = m^2 - n^2$
$b = 2 * m * n$
$c = m^2 + n^2$ because,

$a^2 = m^4 + n^4 – 2 * m^2 * n^2$
$b^2 = 4 * m^2 * n^2$
$c^2 = m^4 + n^4 + 2* m^2 * n^2$

In C

The above method is written in C program as given below:

```
// A C program to generate Pythagorean triplets smaller than a given limit
#include <stdio.h>
#include <math.h>
// Function to generate pythagorean triplets
// smaller than limit
void pythagoreanTriplets(int limit)
{
// triplet: a^2 + b^2 = c^2
int a, b, c=0;
// loop from 2 to max_limitit
int m = 2;
// Limiting c would limit all a, b and c
while (c < limit)
{
// now loop on j from 1 to i-1
for (int n = 1; n < m; ++n)
{
// Evaluate and print triplets using
// the relation between a, b and c
a = m*m - n*n;
b = 2*m*n;
c = m*m + n*n;
if (c > limit)
break;
printf("%d %d %d\n", a, b, c);
}
m++;
}
}
// Driver program
int main()
{
int limit = 20;
pythagoreanTriplets(limit);
return 0;
}
```

In Java

Open in App (https://play.google.com/store/apps/details?id=in.faceprep.app&hl=en)

```
public class Amcat2 {
public static void main(String[] args) {
Scanner sc = new Scanner(in);
int n = sc.nextInt();
int[] arrivingTime = new int[n];
int[] burstTime = new int[n];
int timeQuant;
int[] ans;
for(int i = 0; i < n; i++){
arrivingTime[i] = sc.nextInt();
}
for(int i = 0; i < n; i++)
burstTime[i] = sc.nextInt();
timeQuant = sc.nextInt();
ans = waitingTime(n, arrivingTime, burstTime, timeQuant);
for(int i = 0; i < n; i++)
out.print(ans[i] + " ");
out.println();
}
private static int[] waitingTime(int n, int[] a, int[] b, int q) {
int[] result = new int[n];
int[] processNo = new int[n];
for(int i = 0; i < n; i++)
processNo[i] = i;
b = sort(a,b,processNo);
sort(a);
int max = b[0];
int index = 0;
for(int i = 1; i < n; i++)
if(max < b[i]){
max = b[i];
index = i;
}
int[] c = new int[n];
int time = 0;
for(int i = 0; i < n; i++){
c[i] = b[i];
out.println(a[i] + " " + b[i]);
}
//System.out.println("max : " + max);
while(b[index] != 0){
for(int i = 0; i < n; i++){
if(b[i] == 0)
continue;
if(b[i] >= q){
b[i] -= q;
time += q;
if(b[i] == 0)
result[i] = time - a[i] - c[i];
}else{
time += q - b[i];
b[i] = 0;
result[i] = time - a[i] - c[i];
}
}
}
int temp;
for(int i = 1; i < n; i++){
for(int j = 0; j < i; j++){
if(processNo[i] < processNo[j]){
temp = result[i];
result[i] = result[j];
result[j] = temp;
}
}
}
return result;
}
private static int[] sort(int[] a, int[] b, int[] processNo) {
int temp;
for(int i = 1; i < a.length; i++){
for(int j = 0; j < i; j++){
if(a[i] < a[j]){
temp = a[i];
a[i] = a[j];
a[j] = temp;
temp = b[i];
b[i] = b[j];
b[j] = temp;
temp = processNo[i];
processNo[i] = processNo[j];
processNo[j] = temp;
}
}
}
return b;
}
}
```

Question 5: Write a program to find the sum of contiguous sub array within a one-dimensional array of numbers which has the largest sum.

Explanation:

Simple idea of the Kadane's algorithm is used here. The algorithm is, to look for all positive contiguous segments of the array (max_ending_here is used for this) and keep a track of maximum sum of contiguous segment among all positive segments. max_so_far is used for this. Each time we get a positive sum, compare it with max_so_far and update max_so_far if it is greater than max_so_far.

In C++

```
#include<iostream>
#include<climits>
using namespace std;
int maxSubArraySum(int a[], int size)
{
int max_so_far = INT_MIN, max_ending_here = 0;
for (int i = 0; i < size; i++)
{
max_ending_here = max_ending_here + a[i];
if (max_so_far < max_ending_here)
max_so_far = max_ending_here;
if (max_ending_here < 0)
max_ending_here = 0;
}
return max_so_far;
}
/*Driver program to test maxSubArraySum*/
int main()
{
int a[] = {-2, -3, 4, -1, -2, 1, 5, -3};
int n = sizeof(a)/sizeof(a[0]);
int max_sum = maxSubArraySum(a, n);
cout << “Maximum contiguous sum is ” << max_sum;
return 0;
}
```

Question 6: Write a program for bubble sorting.

Explanation:

The simple way to solve this is to compare two adjacent numbers in an array and swap their position if the second number is greater than the first. Continue this until the output is obtained.

In C:

```
#include <stdio.h>
void swap(int *, int *);
void bubble(int[], int);
int main()
{
int n;
```

Open in App (https://play.google.com/store/apps/details?id=in.faceprep.app&hl=en)

```
printf("Enter the size of n\n");
scanf("%d", &n);
int arr[n];
for (int i = 0; i < n; i++) {
printf("Enter %d Element\n", i);
scanf("%d", &arr[i]);
}
printf("Array before sorting\n");
for (int i = 0; i < n; i++)
{
printf("%d\t", arr[i]);
}
bubble(arr, n);
printf("\nArray after sorting\n");
for (int i = 0; i < n; i++) {
printf("%d\t", arr[i]);
}
printf("\n");
return 0;
}
void bubble(int a[], int n) {
int i, j;
for (i = 0; i < n - 1; i++) {
for (j = 0; j < n - i - 1; j++) {
if (a[j] > a[j + 1]) {
swap(&a[j], &a[j + 1]);
}
}
}
}
void swap(int *a, int *b)
{
*a = *a + *b;
*b = *a - *b;
*a = *a - *b;
}
```

Question 7: Write a program to find the GCD of two Integers.

In C

```
#include
int main()
{
int n1, n2, i, gcd;
printf("Enter two integers: ");
scanf("%d %d", &n1, &n2);
for(i=1; i <= n1 && i <= n2; ++i)
{
// Checks if i is factor of both integers
if(n1%i==0 && n2%i==0)
gcd = i;
}
printf("G.C.D of %d and %d is %d", n1, n2, gcd);
return 0;
}
```

In Java

```
package arrayProject;
import java.util.*;
import java.math.*;
public class GCDPhrSe
{
public static int gcd(int n1,int n2)
{
if(n1==0)
return n2;
if(n2==0)
return n1;
if(n1>n2)
return gcd(n1-n2,n2);
else
return gcd(n2-n1,n1);
}
public static void main(String args[])
{
Scanner sc=new Scanner(in);
int n1,n2;
System.out.println("Enter two no");
n1=sc.nextInt();
n2=sc.nextInt();
int n3=gcd(n1,n2);
System.out.println(n3);
}
}
```

Question 8: Write a program to remove all vowels from a string.

There are two ways to write this program.

In C Using Pointers

```
#include < stdio.h >
#include < stdlib.h >
#include < string.h >
#define TRUE 1
```

Open in App (https://play.google.com/store/apps/details?id=in.faceprep.app&hl=en)

```
# define FALSE 0
int check_vowel(char);
main() {
char string[100], * temp, * pointer, ch, * start;
printf(“Enter a string\n”);
gets(string);
temp = string;
pointer = (char * ) malloc(100);
if (pointer == NULL) {
printf(“Unable to allocate memory.\n”);
exit(EXIT_FAILURE);
}
start = pointer;
while ( * temp) {
ch = * temp;
if (!check_vowel(ch)) { * pointer = ch;
pointer++;
}
temp++;
} * pointer = ‘\0’;
pointer = start;
strcpy(string, pointer); /* If you wish to convert original string */
free(pointer);
printf(“String after removing vowel is \”%s\”\n”, string);
return 0;
}
int check_vowel(char a) {
if (a >= ‘A’ && a <= ‘Z’) a = a + ‘a’ – ‘A’;
if (a == ‘a’ || a == ‘e’ || a == ‘i’ || a == ‘o’ || a == ‘u’) return TRUE;
return FALSE;
}
```

In C Using Switch Case

```
#include <stdio.h>
#include <string.h>
int check_vowel(char);
int main()
{
char s[100], t[100];
int i, j = 0;
printf("Enter a string to delete vowels\n");
gets(s);
for(i = 0; s[i] != '\0'; i++) {
if(check_vowel(s[i]) == 0) { //not a vowel
t[j] = s[i];
j++;
}
}
```

Open in App (https://play.google.com/store/apps/details?id=in.faceprep.app&hl=en)

```
t[j] = '\0';
strcpy(s, t); //We are changing initial string
printf("String after deleting vowels: %s\n", s);
return 0;
}
int check_vowel(char c)
{
switch(c) {
case 'a':
case 'A':
case 'e':
case 'E':
case 'i':
case 'I':
case 'o':
case 'O':
case 'u':
case 'U':
return 1;
default:
return 0;
}
}
```

Question 9: Write a program to return a sorted array from two unsorted array.

In C++

```
#include <bits/stdc++.h>
using namespace std;
// Function to merge array in sorted order
void sortedMerge(int a[], int b[], int res[], int n, int m)
{
// Concatenate two arrays
int i = 0, j = 0, k = 0;
while (i < n)
{
res[k] = a[i];
i += 1;
k += 1;
}
while (j < m) {
res[k] = b[j];
j += 1;
k += 1;
}
// sorting the res array
```

Open in App (https://play.google.com/store/apps/details?id=in.faceprep.app&hl=en)

```
sort(res, res + n + m);
}
// Driver code
int main()
{
int a[] = { 10, 5, 15 };
int b[] = { 20, 3, 2, 12 };
int n = sizeof(a) / sizeof(a[0]);
int m = sizeof(b) / sizeof(b[0]);
// Final merge list
int res[n + m];
sortedMerge(a, b, res, n, m);
cout << “Sorted merged list :”;
for (int i = 0; i < n + m; i++)
cout << ” ” << res[i];
cout << “n”;
return 0;
}
```

In Java

```
public class SortedArray {
public static void main(String[] args) {
int a[]= {8,7,9};
int b[]= {10,5};
sort(a,b);
}
public static void sort(int[] a,int[] b) {
int[] c= new int[a.length+b.length];
for (int i = 0; i < a.length; i++) {
c[i]=a[i];
}
for (int i = 0, j=a.length; i < b.length; i++,j++) {
c[j]=b[i];
}
for (int i = 0; i < c.length; i++) {
for (int j = i+1; j < c.length; j++) {
if(c[i]>c[j]) {
int temp=c[i];
c[i]=c[j];
c[j]=temp;
}
}
}
out.print("Sorted Array : ");
for (int i = 0; i < c.length; i++) {
System.out.print(c[i]+" ");
```

Open in App (https://play.google.com/store/apps/details?id=in.faceprep.app&hl=en)

```
}
}
}
```

Question 10: Write a program to swap two numbers without using third variable,

In C

This question can be solved using three ways as follows:

1) Using + and -

```
#include<stdio.h>
intmain()
{
inta=10, b=20;
printf("Before swap a=%d b=%d",a,b);
a=a+b;//a=30 (10+20)
b=a-b;//b=10 (30-20)
a=a-b;//a=20 (30-10)
printf("\nAfter swap a=%d b=%d",a,b);
return0;
}
```

2) Using * and /

```
#include<stdio.h>
#include<stdlib.h>
intmain()
{
inta=10, b=20;
printf("Before swap a=%d b=%d",a,b);
a=a*b;//a=200 (10*20)
b=a/b;//b=10 (200/20)
a=a/b;//a=20 (200/10)
system("cls");
printf("\nAfter swap a=%d b=%d",a,b);
return0;
}
```

3) Using Bitwise XOR

```
#include <stdio.h>
int main()
{
int x = 10, y = 5;
// Code to swap 'x' (1010) and 'y' (0101)
x = x ^ y;  // x now becomes 15 (1111)
y = x ^ y;  // y becomes 10 (1010)
x = x ^ y;  // x becomes 5 (0101)
printf("After Swapping: x = %d, y = %d", x, y);
return 0;
}
```

Question 11: Write a program to find out sum of digits of given number.

This is a tricky question to solve. The algorithm to solve the question is given below:

1) Take the integer as input.

2) Divide the input integer by 10, obtain its remainder and quotient.

3) Increment the new variable with the remainder obtained in step 2.

4) Repeat the step 2 & 3 with the quotient obtained until the quotient becomes zero.

5) Print the output and exit.

In C

```
/*C program to accept an integer & find the sum of its digits*/
#include <stdio.h>
void main()
{
long num, temp, digit, sum = 0;
printf("Enter the number \n");
scanf("%ld", &num);
temp = num;
while (num > 0)
{
digit = num % 10;
sum = sum + digit;
num /= 10;
}
printf("Given number = %ld\n", temp);
printf("Sum of the digits %ld = %ld\n", temp, sum);
}
```

Question 12: Write a program to print Fibonacci series of given range.

In C

```
#include <stdio.h>
#include <conio.h>
void main()
```

Open in App (https://play.google.com/store/apps/details?id=in.faceprep.app&hl=en)

```
{
long a=1 ,b=1 , c ;
int i , n ;
printf("Enter n: ") ;
scanf("%d" , &n) ;
printf("Fibonacci series is as shown: \n") ;
if(n==1)
printf("% ld \n %ld \n" , a , b) ;
else if(n>1)
{
printf("% ld \n %ld \n" , a , b) ;
c=a+b ;
do
{
printf(" %ld \n" , c) ;
a=b ;
b=c ;
c=a+b ;
}
while(c<=n) ;
}
}
```

Difficult AMCAT Automata Questions

One of the two questions asked in AMCAT Automata is difficult to solve. Here are a few of those questions discussed below:

Question 1: A system that can run multiple concurrent jobs on a single CPU have a process of choosing which task hast to run when, and how to break them up, called "scheduling". The Round-Robin policy for scheduling runs each job for a fixed amount of time before switching to the next job. The waiting time for a job is the total time that it spends waiting to be run. Each job arrives at particular time for scheduling and certain time to run, when a new job arrives, It is scheduled after existing jobs already waiting for CPU time
Given list of job submission, calculate the average waiting time for all jobs using Round-Robin policy.
The input to the function waitingTimeRobin consist of two integer arrays containing job arrival and run times, an integer n representing number of jobs and am integer q representing the fixed amount of time used by Round-Robin policy. The list of job arrival time and run time sorted in ascending order by arrival

Open in App (https://play.google.com/store/apps/details?id=in.faceprep.app&hl=en)

time. For jobs arriving at same time, process them in the order they are found in the arrival array. You can assume that jobs arrive in such a way that CPU is never idle.
The function should return floating point value for the average waiting time which is calculated using round robin policy.
Assume 0<=jobs arrival time < 100 and 0<job run time <100.

In C

```
#include<stdio.h>
int waitingtimerobin(int *job,int *run,int n,int tq)
{
int j,count,time,remain,flag=0;
int wait_time=0,turnaround_time=0,rt[10];
remain=n;
for(count=0;count<n;c++)
{
rt[count]=run[count];
}
for(time=0,count=0;remain!=0;)
{
if(rt[count]<=tq && rt[count]>0)
{
time += rt[count];
rt[count]=0;
flag=1;
}
else if(rt[count]>0)
{
rt[count] -= time_quantum;
time += time_quantum;
}
if(rt[count]==0 && flag==1)
{
remain--;
wait_time += time-job[count]-run[count];
flag=0;
}
if(count==n-1)
count=0;
else if(job[count+1]<=time)
count++;
else
count=0;
}
printf("waiting time %f",wait_time*1.0/n);
return 0;
```

Open in App (https://play.google.com/store/apps/details?id=in.faceprep.app&hl=en)

```
}
int main()
{
int ja[],at[];
int n,tq;
scanf("%d",&n);
for(i=0;i<n;i++)
{
scanf("%d%d", &ja[i],&at[i]);
}
scanf("%d",&tq);
int *cellsptr=ja;
int *cellsptr1=at;
waitingtimerobin(cellsptr,cellsptr1,n,tq);
return 0;
}
```

Question 2: Mooshak the mouse has been placed in a maze.There is a huge chunk of cheese somewhere in the maze. The maze is represented as a two dimensional array of integers, where 0 represents walls.1 repersents paths where mooshak can move and 9 represents the huge chunk of cheese.
Mooshak starts in the top left corner at 0.
Write a method is Path of class Maze Path to determine if Mooshak can reach the huge chunk of cheese. The input to is Path consists of a two dimensional array gnd for the maze matrix. the method should return 1 if there is a path from Mooshak to the cheese.and 0 if not Mooshak is not allowed to leave the maze or climb on walls.
EX: 8 by 8(8*8) matrix maze where Mooshak can get the cheese.
1 0 1 1 1 0 0 1
1 0 0 0 1 1 1 1
1 0 0 0 0 0 0 0
1 0 1 0 9 0 1 1
1 1 1 0 1 0 0 1
1 0 1 0 1 1 0 1
1 0 0 0 0 1 0 1
1 1 1 1 1 1 1 1

Open in App (https://play.google.com/store/apps/details?id=in.faceprep.app&hl=en)

Test Cases:
Case 1:
Input:[[1,1,1,][9,1,1],[0,1,0]]
Expected return value :1
Explanation:
The piece of cheese is placed at(1,0) on the grid Mooshak can move from (0,0) to (1,0) to reach it or can move from (0,0) to (0,1) to (1,1) to (1,0)

Test case 2:
Input:
[[0,0,0],[9,1,1],[0,1,1]]
Expected return value:0
Explanation:
Mooshak cannot move anywhere as there exists a wall right on (0,0)

In C

```
Int isPath(int **grid,int m,int n)
{
if (x,y outside maze) return false
if (x,y is goal) return true
if (x,y not open) return false
mark x,y as part of solution path
if (FIND-PATH(North of x,y) == true) return true
if (FIND-PATH(East of x,y) == true) return true
if (FIND-PATH(South of x,y) == true) return true
if (FIND-PATH(West of x,y) == true) return true
unmark x,y as part of solution path
return false*/
public boolean findPath(int x, int y){
// check x,y are outside maze.
if(x < 0 || x >= mazelength || y < 0 || y >= mazelength )
{
return false;
}
if(maze[x][y] == 9)
{
return true;
}
if(maze[x][y] != 1) return false;   //mark x, y as part of the solution path
if(maze[x][y] == 1)
{
```

Open in App (https://play.google.com/store/apps/details?id=in.faceprep.app&hl=en)

```
maze[x][y] = 3;
}
// move North
if( findPath(x-1,y)){
return true;
}
//move East
if( findPath(x,y+1)) return true;
//move South
if( findPath(x+1,y)) return true;
//move West
if( findPath(x,y-1)) return true;
// unMark x,y as part of the solution.
{
maze[x][y] = 0;
return false;
}
public void printSolution(){
out.println("Final Solution : ");
for(int i=0;i<mazelength;i++){
for(int j=0;j<mazelength;j++){
out.print(" "+maze[i][j]+" ");
}
out.println();
}
}
public static void main(String args[])
{
RatMazeProblem ratMazeProblem = new RatMazeProblem();
out.println(" is Path exist : "+ratMazeProblem.findPath(0,0));
printSolution();
}
}
}
```

Question 3: The Least-Recently-Used(LRU) cache algorithm exists the element from the cache(when it's full) that was least-recently-used. After an element is requested from the cache, it should be added to the cache(if not already there) and considered the most-recently-used element in the cache. Given the maximum size of the cache and a list of integers(to request from the cache), calculate the number of cache misses using the LRU cache algorithm. A cache miss occur when the requested integer does not exist in the cache. Initially, the cache is empty.

The input to the function LruCountMiss shall consist of an integer max_cache_size, an array pages and its length len.
The function should return an integer for the number of cache misses using the LRU cache algorithm.
Assume that the array pages always have pages numbered from 1 to 50.

Test Case 1:
INPUT: 3,[7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0],16
EXPECTED RETURN VALUE: 11

Test Case 2:
INPUT: 2,[2,3,1,3,2,1,4,3,2],9
EXPECTED RETURN VALUE: 8

In C

```
#include<iostream>
using namespace std;
int lruCountMiss(int max_cache_size, int *pages,int len)
{
int miss=0,cache[max_cache_size];
/*variable miss and cache declared*/
for (int i=0;i<max_cache_size;i++)
{
/*this is for clearing value of cache i.e. to empty cache. I used any value here*/
cache[i]=0x87787;
}
for (int i=0;i<len;i++)
{
/*for loop for all the value from list one by one*/
for(int j=0;j<max_cache_size;j++)
{
/*loop to check the cache value, if it matches the value*/
if (pages[i]==cache[j]){ for (int k=i; k<max_cache_size;k++)
{
/*if the value is already present in the cache then shifting values from the present value.*/
cache[i]=cache[i+1];
}
cache[max_cache_size-1]=pages[i];
/*updating the last value of cache*/
break;
}
else if(j==(max_cache_size-1))
{
```

Open in App (https://play.google.com/store/apps/details?id=in.faceprep.app&hl=en)

```
for (int l=0; l<max_cache_size;l++)
{
/*if the value is not present in the cache then shifting values from starting.*/
cache[l]=cache[l+1];
}
cache[max_cache_size-1]=pages[i];
/*updating the last value of cache*/
miss++; } }} return miss;
}
/*returning the Miss*/
int main()
{
/*main function*/
cout<< "We are checking two cases.\n"<<"Case 1:t"<<"Array values are
[7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0] n and cache size= 3; and total values to check are 16: n We got the
miss. ";
int days,pages[]={7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0};
/*array to pass through function*/
int *cellsptr=pages;
/*creating array values to pointer*/
cout<<"Miss =t"<<lruCountMiss(3,cellsptr,16);
//passing to function
cout<<"nnCase 2:t"<<"Array values are [2,3,1,3,2,1,4,3,2] nand cache size= 2; and total values to
check are 9: n We got the miss. ";
int pages2[]={2,3,1,3,2,1,4,3,2};
int *cellsptr2=pages2;
/*creating array values to pointer*/
cout<<"Miss =t"<<lruCountMiss(2,cellsptr2,9);//passing to function return 0;
}
```

Question 4: Write a function to insert an integer into a circular linked _list whose elements are sorted in ascending order smallest to largest. The input to the function insertSortedList is a pointer start to some node in the circular list and an integer n between 0 and 100. Return a pointer to the newly inserted node.

The structure to follow for a node of the circular linked list is_

```
Struct CNode ;
Typedef struct CNode cnode;
Struct CNode
{
Int  value;
Cnode* next;
};
```

Open in App (https://play.google.com/store/apps/details?id=in.faceprep.app&hl=en)

```
Cnode* insertSortedList (cnode* start,int n
{
//WRITE YOUR CODE HERE
}
//FUNCTION SIGNATURE ENDS
```

Test Case 1:
Input: [3->4->6->1->2->^],5
Expected Return Value: [5->6->1->2->3->4->^]

Test Case 2:
Input: [1->2->3->4->5->^],0
Expected Return Value: [0->1->2->3->4->5->^]

Algorithm:

Allocate memory for the newly inserted node and put data in it. Let the pointer to the new node be new_node. After memory allocation, following three cases needs to be handled.

1) Linked List is empty:

a) Since new_node is the only node in CLL, make a self loop.

```
new_node->next = new_node;
```

b) change the head pointer to point to new node.

```
*head_ref = new_node;
```

2) New node is to be inserted just before the head node:

Find out the last node using a loop.

```
while(current->next != *head_ref)
```

```
current = current->next;
```

(b) Change the next of last node.

```
current->next = new_node;
```

(c) Change next of new node to point to head.

```
new_node->next = *head_ref;
```

(d) change the head pointer to point to new node.

```
*head_ref = new_node;
```

3) New node is to be inserted somewhere after the head node:

Locate the node after which new node is to be inserted.

```
while ( current->next!= *head_ref &&

current->next->data < new_node->data)

{   current = current->next;   }
```

(b) Make next of new_node as next of the located pointer

```
new_node->next = current->next;
```

(c) Change the next of the located pointer

```
current->next = new_node;
```

In C

```
#include<stdio.h>
#include<stdlib.h>
/* structure for a node */
struct Node
{
int data;
struct Node *next;
};
/* function to insert a new_node in a list in sorted way.
Note that this function expects a pointer to head node
as this can modify the head of the input linked list */
void sortedInsert(struct Node** head_ref, struct Node* new_node)
{
struct Node* current = *head_ref;
// Case 1 of the above algo
if (current == NULL)
{
new_node->next = new_node;
*head_ref = new_node;
}
// Case 2 of the above algo
else if (current->data >= new_node->data)
{
/* If value is smaller than head's value then
we need to change next of last node */
while(current->next != *head_ref)
current = current->next;
```

Open in App (https://play.google.com/store/apps/details?id=in.faceprep.app&hl=en)

```
current->next = new_node;
new_node->next = *head_ref;
*head_ref = new_node;
}
// Case 3 of the above algo
else
{
/* Locate the node before the point of insertion */
while (current->next!= *head_ref &&
current->next->data < new_node->data)
current = current->next;
new_node->next = current->next;
current->next = new_node;
}
}
/* Function to print nodes in a given linked list */
void printList(struct Node *start)
{
struct Node *temp;
if(start != NULL)
{
temp = start;
printf("\n");
do {
printf("%d ", temp->data);
temp = temp->next;
} while(temp != start);
}
}
/* Driver program to test above functions */
int main()
{
int arr[] = {12, 56, 2, 11, 1, 90};
int list_size, i;
/* start with empty linked list */
struct Node *start = NULL;
struct Node *temp;
/* Create linked list from the array arr[].
Created linked list will be 1->2->11->12->56->90 */
for (i = 0; i< 6; i++)
{
temp = (struct Node *)malloc(sizeof(struct Node));
temp->data = arr[i];
sortedInsert(&start, temp);
}
printList(start);
return 0;
}
```

Open in App (https://play.google.com/store/apps/details?id=in.faceprep.app&hl=en)

These are the most commonly asked AMCAT Automata questions with their answers. Even if the candidate can’t derive an output for the question, it is advisable to try to write the code as much as he can. This is because, even partial answers and getting input carries marks.

All the Best!

Others (https://www.faceprep.in/?ref_type=cat_id&ref_id=5#tag)

General (https://www.faceprep.in/?ref_type=topic_id&ref_id=102#tag)

AMCAT (https://www.faceprep.in/?ref_type=company_id&ref_id=2#tag)

18 Thanks 1 Comment

Thank Comment Share

Praneel Bhatt

Much needed

8 days ago Thank

Open in App (https://play.google.com/store/apps/details?id=in.faceprep.app&hl=en)