**MST -2**

**2nd SEMESTER**

**CS101-Introduction to C Programming**

**Set- A**

**Time allowed: 120 Minutes Max. Marks: 30**

**General Instructions:**

* **All Questions are mandatory**

1. What is the meaning of following statement?

int \*a[100];

1. **Array of integer pointers of size 100**
2. Integer array to integer pointers of size 100
3. Integer array of size 100 to an integer pointer
4. Pointer to an integer array of size 100

2. Default storage class for variable is?

1. register
2. extern
3. static
4. **auto**

3. What is the return type of malloc() or calloc()  
**(A) void \***  
**(B)** Pointer of allocated memory type  
**(C)** void \*\*  
**(D)** int \*

**4.**  Which of the following statements is true?

a) Recursion is always better than iteration

**b) Recursion uses more memory compared to iteration**

c) Recursion uses less memory compared to iteration

d) Iteration is always better and simpler than recursion

5. The elements in the array of the following code are

int array[5] = {5};

a) 5, 5, 5, 5, 5

**b) 5, 0, 0, 0, 0**

c) 5, (garbage), (garbage), (garbage), (garbage)

d) (garbage), (garbage), (garbage), (garbage), 5

6. Predict the output?

#include <stdio.h>

int main()

{

float marks[20]; // Assume that base address of marks is 1100 and size of float is 4 bytes i.e. 32 bits

marks++;

printf("%d", marks); return 0; }

A) 1100

B) 1104

C) 1180

**D) lvalue required**

**7.** What is output of the following code?

#include <stdio.h>

void recursive\_function(int num)

{

if(num == 0)

return;

printf("%d ",num);

recursive\_function(num-1);

}

int main()

{

recursive\_function(5);

return 0;

}

1. **5 4 3 2 1**
2. 5 4 3 2 1 0
3. 1
4. 5

8. Predict the output for following output.

#include<stdio.h>

int main()

{

char \*ptr1 = "Chitkara", \*ptr2;

ptr2=ptr1;

ptr1 = "University";

printf("%s %s\n", ptr1, ptr2);

return 0;

}

1. **University Chitkara**
2. Chitkara University
3. University University
4. Chitkara Chitkara

9. Predict the output

#include <stdio.h>

int fun(int n)

{

if (n == 4)

return n;

else return 2+fun(n-1);

}

int main()

{

printf("%d ", fun(10));

return 0;

}

1. 10
2. **16**
3. 14
4. None of these
5. Identify the correct output for the following code:

int a[2][3] = { {1,2,3}, {4,5,6} };

int j,i;

int sum=0;

for( i = 0; i< 2; i++)

for( j = 0; j < 3; j++ )

if( (i + j) %2 == 0)

sum = sum + a[i][j];

printf(“%d”, sum);

1. **9**
2. 0 1 1 4 4 9 9
3. 12
4. 0 1 4 9

**Coding-1 (5 marks)**

In a game, a string is shown to a user. It asks user to find the frequency of a certain character in the string in a very short time frame and also whether occurrence is even number of times or odd. Write a C code to design the game which finds frequency of a character in a string and checks whether frequency is even or odd.

**Sample input 1**

hello world //String which is to be entered by user

l //character to be searched

**Sample output 1**

l found 3 times in hello world //frequency of character searched

odd // (l occurs 3 times so 3 is odd)

**Sample input 2**

hello world//String which is to be entered by user

u//character to be searched

**Sample output 2**

u found 0 times in hello world //u doesnot occur in hello world

even // 0 is even

**TEST CASES**

**Test case 1**

Input

hello world

;

Output

; found 0 times in hello world

even

**Test case 2**

Input

12 3 4568898iu60

8

Output

8 found 3 times in 12 3 4568898iu60

odd

**Test case 3**

Input

HI!

i

Output

i found 0 times in HI!

even

**Test case 4**

Input

RUthere? Yesyou?

?

Output

? found 2 times in RUthere? Yesyou?

even

**Test case 5**

Input

\*\* .... \*\*

.

Output

. found 4 times in \*\* .... \*\*

even

#include <stdio.h>

#define MAX 100

int main()

{

char str[MAX]={0};

char ch;

int count,i;

scanf("%[^\n]s",str); //read string with spaces

getchar(); // get extra character (enter/return key)

//input character to check frequency

ch=getchar();

//calculate frequency of character

count=0;

for(i=0; str[i]!='\0'; i++)

{

if(str[i]==ch)

count++;

}

printf("%c found %d times in %s\n",ch,count,str);

if (count%2==0)

{

printf("even");

}

else

{

printf ("odd");

}

return 0;

}

**Coding-2(10 marks)**

Write a C program to find the second largest and second smallest number in the one –dimensional array and compute the average of these two numbers. Then check if this average exists in the array or not. If exists then how many times it has occurred in the array.

**Input Format**

The input consists of n integers.

The first line corresponds to n, the number of elements in the array.

The remaining integers correspond to the elements in the array. Assume that the maximum value of n is 20.

**Output Format**

The first line of output must demonstrate the second largest and second smallest respectively.

The second must display the average of these two numbers and next line must contain the count value i.e. 0 (zero) if average does not exist in the array otherwise the count value means how much time it occurs in the array.

Sample Input1:

6

27

66

87

43

59

31

Sample Output1:

66 31

48

0– (As 0 - the average value is not present in an input array)

Sample Input2:

7

12

10

5

8

11

8

4

Sample output2:

11 5

8

2 – (As 8 - the average value is present two times in an input array)

**Input Testcase 1:**

**5**

**388**

**345**

**444**

**416**

**654**

**Output Testcase 1:**

**444 388**

**416**

**1**

**Input Testcase 2:**

**7**

**-122**

**198**

**675**

**-329**

**568**

**321**

**432**

**Output Testcase 2:**

**568 -122**

**223**

**0**

**Input Testcase 3:**

**10**

**45**

**65**

**34**

**87**

**12**

**8**

**40**

**8**

**5**

**15**

**Output Testcase3:**

**65 8**

**36**

**0**

**Input Testcase 4:**

**8**

**12**

**10**

**5**

**8**

**11**

**8**

**4**

**8**

**Output Testcase 4:**

**11 5**

**8**

**3**

**Input Testcase 5:**

**5**

**67**

**22**

**11**

**33**

**45**

**Output Testcase 5:**

**45 22**

**33**

**1**

Solution:

#include <stdio.h>

void read(int \*number,int n);

int returnAverage(int \*number,int n);

int findAverage(int \*number,int n,int avg);

int main ()

{

int number[30];

int n;

scanf("%d", &n);

read(number,n);

int avg=returnAverage(number,n);

printf("%d\n",avg);

int counter=findAverage(number,n,avg);

printf("%d\n",counter);

return 0; }

void read(int \*number,int n)

{

int i;

for (i = 0; i < n; i++)

scanf("%d", &number[i]);

}

int returnAverage(int \*number,int n)

{

int i, j,a,average;

for (i = 0; i < n; i++)

{

for (j = i + 1; j < n; j++)

{

if (number[i] < number[j])

{

a = number[i];

number[i] = number[j];

number[j] = a;

}

}

}

printf("%d\t%d\n",number[1],number[n-2]);

average = (number[1] + number[n - 2]) / 2;

return average;

}

int findAverage(int \*number,int n,int avg)

{

int counter = 0,i;

for (i = 0; i < n; ++i)

{

if (avg == number[i])

{

counter++;

}

}

return counter;

}