SRN | | | | | | | | | | | | |

PES University, Bengaluru

(Established under Karnataka Act No. 16 of 2013)

UE21CS141B

JULY 2022: END SEMESTER ASSESSMENT B.TECH II SEMESTER

UE21CS141B – PROBLEM SOLVING WITH C

Time: 3 Hrs | Answer All Questions | Max Marks: 100

Q1	a)	Mention the phases involved in the Program development Life cycle of a C Program. **Solution: // Any four phases – 4 marks** Editing, Pre-processing, Compiling, Linking, Loading	4 M

	b)	The data file mobiles.txt has the details (**Model_id, Price in Rs and the Number of quantities available**) of 1000 Mobiles. Sample data is as below. A person named Hrithik would like to buy mobile for his sister with the budget of Rs. 20000. Write the C code to display the IDs of Mobile phones whose price is less than or equal to the budget amount.
		Every column in a row is separated by a tab. Use **redirection operator** to provide data file as an input to the executable.
		Hint: a.exe < mobiles.txt OR ./a.out < mobiles.txt
		1 6499 650
		10 7999 80
		3 8999 707
		6 11990 1199
		5 10990 1099
		4 9990 676
		7 19990 1999
		8 36990 3699
		9 36990 3699
		2 25990 2599
		…
		Solution:
		`#include<stdio.h>`
		`int main()`
		`{`
		`int i = 0; int id; int price; int num_quant; // variable declarations 1 mark`
		`printf("reading from the data file\n");`
		`while(i <= 1000) // any loop, 1 mark`
		`{ scanf("%d %d %d",&id,&price,&num_quant); // 1 mark`
		`if(price<20000) // 1 mark`
		`{ printf("Model id is %d\n",id); // 1 mark }`
		`i++; // 1 mark`
		`}`
		`return 0;`
		`}`
			6 M

	c)	Briefly explain any two differences between while and do-while looping constructs in C. **Solution: 2 points for each – 4 marks** while: top tested. Body executes zero or more times bottom tested. Body executes one or more times	4 M

	d)	What does the below code do? Also list the keywords in the same code.
		`#include<stdio.h>`
		`int main()`
			6 M
			(4+2)

```
{       int sum = 0;    int j;
        for(j = 0 ; j < 100 ; j++)
        {               sum+=j;         }
        printf("%d",sum);
        return 0;
}
```

Solution:

Prints the sum of all the integers from 0 to 99 and prints the sum. – 2 marks

Keywords – for, int, include, stdio, printf, main -- any 4, 4 marks

Q2 a) Provide a brief note on below commands in GDB. — 4 M **(2+2)**

i) list ii) clear

Solution:

// Any two valid points, 2 marks

list OR l: Used to print lines from a source file. By default, ten lines are printed

- list line_num : Print lines centered around that specified line number
- list func: Print lines centered around the beginning of function func.
- list: Print more lines.
- list -: Print lines just before the lines last printed.
- list first, last: print lines from first to last
- list ,last: Print lines ending with last

// Any two valid points, 2 marks

clear: Delete any breakpoints set at the specified location .

- clear line_num: all breakpoints in that line are cleared.
- clear file_name: breakpoints at beginning of function are cleared
- clear: Delete any breakpoints at the next instruction to be executed in the selected stack frame.

b) Given the client code, write the implementation of separate function for separating the even and odd elements from the given array and store it in respective arrays. — 6 M

```
#include<stdio.h>
int main()
{
        int a[] = {7,4,2,7,6,12,9};         int n = sizeof(a)/sizeof(*a);
        int a_even[100];        int a_odd[100];
        int count_even = 0;     int count_odd = 0;
        separate(a,n,a_even,a_odd,&count_even, &count_odd);
        return 0;
}
```

Solution:

```
void separate(int *a, int n, int *aeven, int *aodd, int *ceven, int *codd)  // 2 marks
{
        for(int i = 0 ; i < n; i++) // 1 mark
        {
                if(a[i] %2 == 0)        // 1 mark
                {
                        (*ceven)++;             // 1 mark
                        aeven[*ceven-1] = a[i]; // 1 mark
                }
                else
                {
                        (*codd)++;
                        aodd[*codd-1] = a[i];
                }
        }
}
```

c) Find the output of below code snippet when executed separately. — 4 M

		i) char *a = "immunization and health record";
		a[1] = 'M';
		printf("%s",a);
		ii) int a[10] = {7,4,9,0,12,87}
		printf("%d %d", a[6], a[1]);
		Solution: // 2 marks each
		i) Runtime Error
		ii) 0 4	

	d)	Write the user defined recursive function to find the length of the string. Also test this function in the client code.
		Solution:
		int my_strlen(char *str)
		{
		if (!(*str)) // 1mark
		return 0; // 1 mark
		else
		return 1+my_strlen(++str); // 2 mark
		}
		int main() // 2 marks for client code
		{
		char mystr[ ] = "pes";
		printf("Length is %d\n", my_strlen(mystr));
		return 0;
		}
			6 M

Q3	a)	List any two functions related to dynamic memory allocation using C. Explain anyone with a code snippet. **Solution:** malloc, calloc, free, realloc – any two, 1 mark Explanation – 2 marks coding statement – 1 mark	4 M

	b)	Given the structure definitions, alias for it and the client code, write the definition of insert_list function so that it creates the ordered list in ascending order for a given n elements.
		struct node
		{ int info; struct node *link; };
		typedef struct node NODE_T;
		
		struct mylist
		{ NODE_T* head; };
		typedef struct mylist MYLIST_T;
		
		#include<stdio.h>
		int main()
		{ MYLIST_T mylist;
		mylist.head=NULL;
		printf("enter the number of elements u want to insert\n");
		int n,ele; int i;
		scanf("%d",&n);
		for(i = 0;i<n;i++)
		{ printf("enter the element-->");
		scanf("%d",&ele);
		insert_list(&mylist,ele);
		}
		printf("Elements of the list are\n");
			6 M

```
        display_list(&mylist);
        free_list(&mylist);
        return 0;
}
```

Solution:

```
void insert_list(MYLIST_T* p_list,int ele)
{
        NODE_T* temp = (NODE_T*)malloc(sizeof(NODE_T));   // create node – 1 mark
        temp -> info = ele;
        temp -> link = NULL;
        if (p_list -> head == NULL) p_list->head = temp;
        NODE_T* present = p_list->head;          NODE_T* prev = NULL;
        while(present != NULL && present->info < ele) // traversal  2 marks
        {       prev = present;
                present = present->link;
        }
        if(prev == NULL)                // 1mark
        {       temp->link = present;     // 1 mark
                p_list->head = temp;  // insert at the beginning
        }
      else
        {       temp->link = present;
                prev -> link =temp;  // insert in between or at the end  // 1 mark
        }
}
```

c) In the below code, if the address of a[0][0] is 6000 and the size of integer is 4 bytes, find the address of a[1][2]. Also mention what gets printed? — 4 M

```
#include<stdlib.h>
int main()
{
      int a[][2] = {3,5,6,7,8,1,99,66,33,15,36};
      printf("%d",a[1][2]);       return 0;
}
```

Solution: // 2 marks each

6016

8

d) — 6 M

```
struct Car
{       int year;       char company[100];      char color[100];    };
typedef struct Car CAR;
```

Given the above structure definition and alias for it, write the C statements for the following.

i) Create a pointer to structure.

ii) Allocate memory for all the members of the structure dynamically

iii) Assign values to all the members of the structure through this pointer.

Solution: // 2 each

i) CAR *c;

ii) c = (CAR*)malloc(sizeof(CAR));

iii) c->year = 1990; strcpy(c->company, "benz"); strcpy(c->color,"white");

Q4 a) Below code must copy all the data from src.txt to dest.txt. Fill up the blank spaces to do the — 4 M

same.

```
#include<stdio.h>
int main()
{       FILE *fr = fopen("src.txt","r");
        FILE *fw = fopen ("dest.txt","   _____________  ");
        if(fr == NULL)
                printf("cannot read file");
        else
        {
                char c;
                while(_____________________________)
                {
                        fputc(c,fw);
                }
        }
        return 0;
}
```

// each 2 marks

w

(c = fgetc(fr)) != EOF

b) Implement Binary search using iterative method on an array of 100 integer elements which are stored in descending order. Handle both successful and unsuccessful search. — 6 M

Given the array, int a[ ] = {100,98,76,54,44,43,42,40,31,30};

Write only the function definition. Client code is not a requirement

Solution:

```
int mysearch(int a[],int low,int high,int key)
{
        int pos = -1;
        int found = 0;
        while(low<=high && found ==0)
        //while(low<=high && pos != mid)
        // while(low<=high && pos == -1)      // low <= high    1 mark
        {
                int mid = (low+high)/2;          // 1 mark
                if(a[mid]==key)        // 1 mark
                        {
                        pos = mid; found = 1;
                        }
                else if(key>a[mid])        // 1 mark
                        high = mid-1;     // 1 mark
                else
                        low = mid+1;       // 1 mark
        }
        return pos;
}
```

c) Write a neat diagram to depict the below code and also get the output of this code. — 4 M **(2+2)**

```
#include<stdio.h>
int main()
{
        int data[10] = {3,8,1,4};
        int *datap[12];
        datap[11] = &data[2];
        printf("%d\t",*datap[11]);
```

```
        datap[11] = &data[9];
        printf("%d\t",*(datap[11]));
        return 0;
}
```

Solution: //Diagram 2 marks

1 0 // output 2 marks

d) Given a dataset containing the details of trains from Bangalore to Mumbai(sample given below), sort it based on the availability and based on the cost separately using selection sort algorithm.. 6 M

PNR	Train_name	Source City	Destination city	Cost	Availability
22416	AndhraPradesh Express	Bangalore	Mumbai	1000	7
12724	AndhraPradeshExpress	Bangalore	Mumbai	500	56
12707	Andhra Pradesh Sampark Kranti	Bangalore	Mumbai	800	10
15909	Abadh Assam Express	Bangalore	Mumbai	1000	67
18242	Abkp Durg Passenger E	Bangalore	Mumbai	800	3
11266	Abkp Jbp Express	Bangalore	Mumbai	600	6
58702	Abkp Sdl Passenger	Bangalore	Mumbai	750	4
54703	Abs Ju Passengr	Bangalore	Mumbai	850	80
7509	Adb Qln Special	Bangalore	Mumbai	900	66
9416	Adi Madgaon Special	Bangalore	Mumbai	450	48

…

The new type struct Train is used to store the detail of each train. The array train_arr contains the data from the dataset. Sort the data in the array of structures as per the requirement in the client code. n is the number of train details from the file.

Include the definitions of compare_availability and compare_cost functions

```
struct Train
{
        char name[100];
        int cost;
        int availability_count;
};
typedef struct Trian TRAIN;
int main()
{
        TRAIN train_arr[10000];
        // Code to read the contents from the datafile and storing those details in the array of structure is available here. Please DO NOT add that code here
        int ch;
        printf("enter the choice.\n1. sort on availability\n2. sort on cost\n");
        scanf("%d",&ch);
        switch(ch)
        {
                case 1: sort(train_arr, n, compare_availability); break;
                case 2: sort(train_arr, n, compare_cost); break;
                default: printf("exiting from the program"); break;
        }
        return 0;
}
void selection_sort(TRAIN *t, int n, int (*com)(TRAIN*,TRAIN*))
{
        // fill this implementation
}
```

Solution:

```
void selection_sort(TRAIN *t, int n, int (*com)(TRAIN*,TRAIN*))
{
        int i,pos,j;
        for(i = 0;i<n-1;i++)                    // 2 loops 1 mark
        {
                pos = i;                        // 1 mark
                for(j = i+1;j<n;j++)
                {
                        if(com(t[pos],t[j]))            // 1 mark
                        {
                                pos = j;        // 1 mark
                        }
                }
                if(pos != i)
                {                                       // swap 1 mark
                   TRAIN temp = t[pos];
                    t[pos] = t[i];
                    t[i] = temp;
                }
        }

}
// 1 mark for any one definition out of two functions
int compare_ availability(TRAIN s1,TRAIN s2)
        {       return s1. availability_count > s2. availability_count;     }
int compare_cost(TRAIN s1,TRAIN s2)
        {       return s1.cost  > s2.cost;     }
```

Q5 a) Mention any four pre-defined macros in C. — 4M

Solution: Any 4, 4 marks

__LINE__ __TIME__ __FILE__ __STDC__ __unix __MINGW32

b) i) Describe any two differences between structure and unions in C. — 6 M **(2+4)**

ii) What gets printed when the below code is compiled and executed?

Assumption: Size of the integer is 4 bytes and no alignment/optimization is done

```
#include<stdio.h>
#include<stddef.h>
union A
{       int x;     int y[2];          int z;     };
int main()
{
        union A a1;         a1.x = 100;
        printf("%d %d %lu\n",z, a1.y[0],offsetof(union A,y[0]), offsetof(union A,y[1]) );
        return 0;
}
```

Solution:

i) Any two differences 2 marks

Structure:

1. The keyword struct is used to define a
2. When a variable is associated with a structure, memory is allocated to each member of the structure. The size of the structure is greater than or equal to the sum of its members.
3. Each member within a structure is assigned unique storage area of location
4. In Structure, the address of each member will be in ascending order. This indicates that memory for each member will start at different offset values.
5. In Structure, altering the value of a member will not affect other members of the

		structure. 6. All members can be accessed at a time. **Union:** 1. The keyword union is used to define a union. 2. When a variable is associated with a union, memory is allocated by considering the size of the largest So, the size of a union is equal to the size of its largest member. 3. Memory allocated for union is shared by individual members of 4. For unions, the address is same for all the members of a union. This indicates that every member begins at the same offset. 5. Altering the value of any of the member will alter other member values. 6. In Unions, only one member can be accessed at a time **ii)100 100 0 4 // 1 mark each**	
	c)	**Say True or False.** i) The constants defined with enum are automatically assigned values if no value specified. ii) Arithmetic operators are allowed on enum constants iii) Enum constants can be non- unique in their scope. iv) Storing the symbol of one enum in another enum variable is valid. **Solution:** True True False True	4M

	d)	Write a C code to find whether the strings passed in the command line are having vowels within it or not(consider the small case only).
		Solution:
		int main(int argc, char* argv[])
		{
		for(int i = 1 ; i < argc; i++) // loop 1 mark
		{
		int j = 0;
		int isvowel = 0;
		while(argv[i][j] != '\0' && isvowel == 0) // inner loop 1 mark
		{
		if(argv[i][j] == 'a' || argv[i][j] == 'e'||argv[i][j] == 'i'||argv[i][j] == 'o'||argv[i][j] == 'u') // condition 2 mark
		isvowel = 1;
		j++; // 1mark
		}
		if(isvowel == 1) // 1mark
		printf("%s has vowel within it\n",argv[i]);
		}
		return 0;
		}
		}
			6 M