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

PES University, Bengaluru

(Established under Karnataka Act No. 16 of 2013)

UE20CS151

JULY 2021: END SEMESTER ASSESSMENT B.TECH II SEMESTER

UE20CS151 – PROBLEM SOLVING WITH C

Time: 3 Hrs | Scheme & Solution | Max Marks: 100

Q1	a)	Mention the four types of errors that may occur during the C program development. **Solution: 1 mark each. Any four** Compiletime Error, Linktime Error, Runtime Error, Logical Error	4 M

	b)	Trace the output of below C code executions?
		i) #include<stdio.h>
		int main()
		{ int a = 12;
		for(;;)
		{ printf("%d",a);
		switch(a)
		{ case 1: break; case 2: break; case 12: a = 1; break; }
		}
		return 0;
		}
		ii) #include<stdio.h>
		int main()
		{ int a = 12; float c = 3.5; printf("%f",a*c); return 0; }
		iii) #include<stdio.h>
		int main()
		{ printf("ALL","THE","BEST"); return 0; }
		Solution: 2 mark each
		i) 12 and then **infinite loop** with 1 as the output.
		ii) 42.000000
		iii) ALL
			6 M

	c)	Separate the keyword/s and the variable/s from the set of identifiers given below. global do int _val **Solution: // 1 mark each.** Keywords → do, int Variables → global, _val	4 M
	d)	Consider the input data file having the details of Mobiles (**Model_id, Phone_type, Number of units sold and the total amount received for each model_id**) in 12 rows. Model id and the Number of units sold is an integer, Phone_type represents a character to denote the Smart phone(S) and the Feature Phone(F). Write a C code to find the price of each Feature phone given the total amount received for each model for those many units sold. Display the Price of each feature phone along with the model_id where the price of each phone is a floating point value. Also, display the number of feature phones in the data given.Input redirection operator is used to provide this data file as an input to the executable of the C code. 4 S 6819 204572 12 F 371 111112 1 S 339 12342 3 S 194 12345 2 S 413 65647	6 M

```
11      S       227     78346
7       F       788     87356
8       F       1208    333572
9       S       125     676763
5       F       347     463539
10      F       330     57643
6       F       192     437452
```

Solution:

```
int main()
{
        int i = 0;  int count = 0;
        int id, units, amount;
        char type;      // variable declarations 1 mark. One variable must be of char type
        while(i<12) // looping 1 mark
        {
                scanf("%d %c %d %d", &id, &type, &units, &amount);  // 1 mark
                if(type == 'F')     // 1 mark
                {       printf("price of %d feature phone is %f\n",id, ((float)amount/units));
                                // type casting 1 mark
                        count++;   //  1 mark for incrementing loop variable and count
                }
                i++;
        }
        printf("\nNumber of feature phones is %d",count);
        return 0;
}
```

Q2 a) Given the client code, fill the implementation of avoid_vowels function to print only the consonants in the string. 4 M

```
#include<stdio.h>
int main()
{
        char a[ ] = "immunization and health record";
        avoid_vowels(a);
        return 0;
}
void avoid_vowels(char str[])
{
        // Fill this implementation
}
```

Solution:

```
void avoid_vowels(char str[])
{
        while(*str)    // any loop 1 mark
        {
                if(*str != 'a' && *str != 'e' && *str != 'i' && *str != 'o' && *str != 'u')   // if condition
using logical operator *&& 2 marks
                        printf("%c",*str);
                str++;          // 1 mark
        }
}
```

b) Given the client code, fill the implementations of Merge_arrays and Display_array. 6 M

Merge_arrays function must merge the contents of the two arrays passed as the argument a1

and a2. The output must be stored in a3.
Display_array function must display all the elements of the array.

```
#include<stdio.h>
void Display_array(int *a,int n);
void Merge_arrays(int *a1,int *a2,int n1,int n2,int *new);
int main()
{
        int a1[ ] =  {23,66,10,37,29};          int a2[ ] = {20,17,49};
        int n1 = sizeof(a1)/sizeof(*a1);        int n2 = sizeof(a2)/sizeof(*a2);
        int a3[1000];     Merge_arrays(a1,a2,n1,n2,a3);     Display_array(a3,n1+n2);
        return 0;
}
void Merge_arrays(int *a1,int *a2,int n1,int n2,int *new)
{       // Fill this implementation        }
void Display_array(int *a,int n)
{       Fill this implementation      }
```

Solution: 4 marks for merge arrays function and 2 marks for display function

```
void Merge_arrays(int *a1,int *a2,int n1,int n2,int *new)
{
        int i,j;
        for(i = 0;I < n1;i++)           // Any two loops, 2 marks. But it cannot be nested.
                new[i] = a1[i];         // 1 mark
        for(j = 0; j < n2; j++)
                new[i+j] = a2[j];       // 1 mark
}
void Display_array(int *a,int n)
{
        int i;
        for(i = 0;i < n;i++)            // loop 1 mark
                printf("%d\t",a[i]);            // printing  1 mark
}
```

c) Write a note on below commands in gdb — 4 M

i) break ii) list

Solution: 2 marks each. No code is required. If code there, max marks allocation is only 2 for each

i) break: Makes the program pause whenever a certain point in the program is reached. Can be specified as below

break line_num

break function_name

break address_of_instruction

break if condition

break with no arguments: Sets the breakpoint at the next instruction to be executed in the selected stack frame

ii) list: Used to print lines from a source file. By default, ten lines are printed. Can be used in below ways.

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

Usage of show listsize and setting listsize

	d)	i) Fill up the blank space using the pointer notation to get the expected output.
		Expected output: 60
		#include<stdio.h>
		int main()
		{ int a[][2] = {34,55,11,17,20,60}; printf("%d", _____); return 0; }
		
		ii) What is the output of below code?
		#include<stdio.h>
		int main()
		{ char list_of_friends[] = {"THAKSH", "INCHARA", "AADIT", "AADHYA"};
		printf("%s",list_of_friends[0]); return 0;
		}
		
		iii) Find the output of the below recursive code.
		#include<stdio.h>
		int get_what(int n1,int n2);
		int main()
		{ int n1 = 12; int n2 = 10; printf("%d",get_what(n1,n2)); return 0; }
		int get_what(int n1,int n2)
		{
		if(n1 == 0 || n2 == 0) return n2;
		else { return get_what(n2-2,n1-1); }
		}
		iv) Name the two types of lines required to complete the rules of the make file creation.
		Solution:
		i) *(*(a+2)+1) // 1 mark
		If used array notation like a[2][1] OR (*(a+2))[1] OR a[0][5] OR a[0][5] , no marks
		ii) Compiletime Error
		iii) -2 // 2 marks
		iv) Dependency line and Action line. // 1 mark each. Total 2 marks
			6 M
			1+1
			2+2

Q3	a)	List any four characteristics/properties of structures in C. **Solution: Any valid 4 points, 4 marks** • Contains one or more components(homogeneous or heterogeneous) – Generally known as data members. These are named ones. • Order of fields and the total size of a variable of that type is decided when the new type is created • Size of a structure depends on implementation. Memory allocation would be at least equal to the sum of the sizes of all the data members in a structure. Offset is decided at compile time. • Compatible structures may be assigned to each other.	4 M
	b)	Complete the function definition of extract_data_display to segregate the even and odd numbers from the given linked list and copy those elements to respective arrays. The function must also print both the arrays. The client code is as below. #include<stdio.h> #include<stdlib.h> struct node { int data; struct node* link; }; typedef struct node NODE; struct list { NODE *head; }; typedef struct list LIST; void extract_data_display(LIST* li,int *even,int *odd); int main()	6 M

```
{
        NODE *n = (NODE*) malloc(sizeof(NODE));
        n->data = 40;           n->link = (NODE*) malloc(sizeof(NODE));
        n->link->data = 33;     n->link->link = (NODE*) malloc(sizeof(NODE));
        n->link->link->data = 25;       n->link->link->link = (NODE*) malloc(sizeof(NODE));
        n->link->link->link->data = 88;  n->link->link->link->link = NULL;
        LIST *li;       li->head = n;   int odd[100];   int even[100];
        extract_data_display(li,even,odd);
        return 0;
}
void extract_data_display(LIST *li,int *even,int *odd)
{   // Fill this implementation    }
```

Solution:

```
void extract_data_display(LIST *li,int *even,int *odd)
{
        int i=0,j=0;
        if(li->head == NULL)
                printf("no elements in the list to display\n");
        else
        {
                NODE *n = li->head;
                while(n != NULL)        // loop 1 mark
                {
                        if(n->data % 2 == 0)    // 1 mark
                        {
                                even[i] = n->data;   // 1 mark
                                i++;            // 1 mark
                        }
                        else
                        {
                                odd[j] = n->data;
                                j++;
                        }
                        n = n->link;
                }
                int k;
                printf("Even elements in the list are\n"); // printing both the arrays 2 marks
                for(k = 0;k < i;k++)
                        printf("%d\t",even[k]);
                printf("\n");
                printf("Odd elements in the list are\n");
                for(k = 0;k < j;k++)
                        printf("%d\t",odd[k]);
                printf("\n");
        }
}
```

c) Write the output of below C code execution. 4 M

```
#include<stdlib.h>
#include<stdio.h>
int main()
{       int *p = (int*)calloc(5,sizeof(int));           *p = 10;
        printf("%d\n",*p);          *(p+3) = 30;
```

```
        printf("%d\n",*p);      p = (int*)realloc(p,8*sizeof(int));     p = p+3;
        printf("%d\n",*p);      p = p+4;
        printf("%d\n",*(p-1));  return 0;
}
```

Solution: 1 mark each

10
10
30
Undefined value

d) The company called SOOKSHM_LIFE_STYLE is launching new collections of kids apparels. The details to be shown to customers by the owner of the company are captured in the structure declaration given below. Implement a read function to read the details of 50 products. Include a display function definition to display the product_name and safety_warning only if the material type is "cotton". Test these functions in the client code. 6 M

```
typedef struct Sookshm_clothing
{       int product_id;  char product_type[10];    char product_name[10]; char care_info[100];
        char occasion[20];      char material_type[100];char spl_feature[100];    char
safety_warning[100];
}SOOKSHM;
```

Solution:

```
void read(SOOKSHM[],int n);
void display(SOOKSHM[],int n);
int main()
{
        SOOKSHM s[50];  // 1 mark for client code. creation of array of structures
        printf("enter the details\n");
        read(s,50);
        printf("Displaying the details of products if material type is cotton\n");
        display(s,50);
        return 0;
}
void read(SOOKSHM s[],int n)
{
        int i;
        for(i = 0;i < n;i++)        // any loop 1 mark
        {
                printf("%d: product: ",i+1);
                scanf("%d %s %s %s %s %s %s %s",&s[i].product_id, s[i].product_type,
s[i].product_name, s[i].care_info, s[i].occasion, s[i].material_type, s[i].spl_feature,
s[i].safety_warning);
                // 1 mark: & for int type and scanf
        }
}
void display(SOOKSHM s[],int n)
{       int i;
        for(i = 0;i < n;i++)   // any loop 1 mark
        {
                if(!strcmp(s[i].material_type,"cotton"))   // 1 mark for strcmp
                        printf("%d %s %s %s %s %s %s %s\n",s[i].product_id,
s[i].product_type,  s[i].product_name,
s[i].care_info,s[i].occasion,s[i].material_type,s[i].spl_feature,s[i].safety_warning);
                        // 1 mark for printf  and using .with array of structures.
        }
}
```

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

Q4 a) Write the pictorial representation(diagram) for the code snippet in bold. **4 M**

```
int a[ ] = {23,66,15,17,11,99};
int *ap[10];
int n  = sizeof(a)/sizeof(*a);
int i;
for(i = 0; i < n; i++)
{           ap[i] = &a[i];            }
for(i = 0; i < n; i++)
{           printf("%d\t",*ap[i]);      }
```

Solution:

b) The brand SOOKSHM has launched new collections for kids whose data is available in sookshm.txt. The sample from the data file is as below. Write a C code to find all the products and its web links if the price of the product is greater than 1000. Write these details to a new file. **6 M**

sookshm - Notepad

File Edit Format View Help

Product_Name	Fit_Type	Sleeve_Type	Material_Type	Web_link	Price
SKSHM-TshirtsBoys000-1-2Yrs	Regular fit	Halfsleeves	Cotton	https://i.postimg.cc/s24Lg6bw/00.jpg	949
SKSHM-TshirtsBoys001-2-3Yrs	Slim fit	Fullsleeves	Cotton	https://i.postimg.cc/s24Lg6bw/03.jpg	949
SKSHM-TshirtsBoys001-3-4Yrs	Regular fit	Halfsleeves	Cotton	https://i.postimg.cc/s24Lg6bw/02.jpg	999
SKSHM-TshirtsBoys001-4-5Yrs	Slim fit	Halfsleeves	Cotton	https://i.postimg.cc/s24Lg6bw/07.jpg	1099
SKSHM-TshirtsBoys001-5-6Yrs	Regular fit	Halfsleeves	Cotton	https://i.postimg.cc/s24Lg6bw/01.jpg	1099
SKSHM-TshirtsBoys001-6-7Yrs	Slim fit	Halfsleeves	Cotton	https://i.postimg.cc/s24Lg6bw/04.jpg	579
SKSHM-TshirtsGirls000-1-2Yrs	Regular fit	Halfsleeves	Cotton	https://i.postimg.cc/59DWGGJC/08.jpg	949
SKSHM-TshirtsGirls001-2-3Yrs	Slim fit	Fullsleeves	Cotton	https://i.postimg.cc/59DWGGJC/10.jpg	949
SKSHM-TshirtsGirls001-3-4Yrs	Regular fit	Fullsleeves	Cotton	https://i.postimg.cc/59DWGGJC/7.jpg	999
SKSHM-TshirtsGirls001-4-5Yrs	Slim fit	Fullsleeves	Cotton	https://i.postimg.cc/59DWGGJC/9.jpg	1099
SKSHM-TshirtsGirls001-5-6Yrs	Regular fit	Halfsleeves	Cotton	https://i.postimg.cc/59DWGGJC/8.jpg	1099
SKSHM-TshirtsGirls001-6-7Yrs	Slim fit	Halfsleeves	Cotton	https://i.postimg.cc/59DWGGJC/6.jpg	579

Solution:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int main()
{
        FILE *fpr = fopen("sookshm.txt","r");  // 1 mark for opening the files
        FILE *fpw = fopen("sookshm_above_1000.txt","w");
        char line[10000];
        char *prod_name;
        char *web_link;
        int price;
        if(fpr == NULL || fpw == NULL) printf("cannot open the file\n");
        else
        {
        fgets(line,10000,fpr);
        while(fgets(line,10000,fpr) != NULL)   // looping 1 mark
        {
                prod_name = strtok(line,"\t");  // 1 mark
                int i = 0;
                while(i<3)    // 1 mark for loop to skip middle three columns
                {
                        strtok(NULL,"\t") ;                    i++;
                }
                web_link = strtok(NULL,"\t");
```

```
                    price = atoi(strtok(NULL,"\t")); // atoi 1 mark
                    if(price > 1000)
                    {
                    fprintf(fpw,"%s %s\n",prod_name,web_link);
                                        // 1 mark for if condition with display
                    }
          }
          fclose(fpr);
          fclose(fpw);
          }
          return 0;
}
```

c) Brief about Error handling in C using errno and strerror. Write the C code snippet for the same. 4 M

Solution: 2 marks for theory and 2 marks for code

errno: Its a global variable indicating the error occurred during any function call and defined in the header file errno.h.

strerror():returns a pointer to the textual representation of the current errno value. Syntax: char *strerror (int errnum) //errnum is the error number (errno).Defined in string.h

```
#include <stdio.h>
#include <errno.h>
#include<string.h>
int main ()
{
   FILE *fp;
   fp = fopen ("File.txt", "r");  // File with this name doesn't exist
   printf("Value of errno: %d %s\n ", errno, strerror(errno));
   return 0;
}
```

d) Implement Binary search on an array of 10 integer elements which are 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};

Solution: Function definition is not compulsory. Recursive or iterative , any implementation is fine

```
int main()
{
          int a[] = {100,98,76,54,44,43,42,40,31,30};
          int key; int n; int i;
          printf("enter the element to be searched\n");
          scanf("%d",&key);
          n = sizeof(a)/sizeof(*a);
          int res = mysearch(a,0,n,key);
          if(res == -1)                    // client code 1 mark
                    printf("not found");
          else
                    printf("found at %d\n",res);
}
int mysearch(int a[],int low,int high,int key)
{
         if(low > high)
                    return -1;     // 1 mark
          else
          {
                    int mid = (low+high)/2;        // 1 mark
                    if(a[mid]==key)
                    {
                              return mid;    // 1 mark for if and return
```

```
                    }
                    else if(key>a[mid])        // 1 mark
                              return mysearch(a,low,mid-1,key); // recursion 1 mark
                    else
                              return mysearch(a,mid+1,high,key);
          }

}
```

Q5 a) List the four types of storage classes in C and explain any two with an example code snippet. 10M

Solution: list – 1 mark each

auto, static, extern, register

Explanation with code snippet for any two types: 3 each. (2 for briefing. 1 for code)

auto: A variable declared inside a function without any storage class specification is by default an automatic variable. They are created when a function is called and are destroyed automatically when the function execution is completed. Automatic variables can also be called local variables because they are local to a function. By default, they are assigned to undefined values.

```
int main()
{
int i=90;// by default auto because defined inside a function main()
auto float j=67.5;        printf("%d %f\n",i,j); return 0;
}
int f1()
{          int a;     // by default auto because declared inside a function f1()       }
```

extern: The extern keyword is used before a variable to inform the compiler that the variable is declared somewhere else. The extern declaration does not allocate storage for variables. All functions are of type extern. The default initial value of external integral type is 0 otherwise null.

```
int main(){
extern int i;        // Information saying declaration exist somewhere else and make it available during linking
// if u comment this line, it throws an error: i undeclared
printf("%d\n",i);
}
int i=23;
```

static: A static variable tells the compiler to persist the variable until the end of program. Instead of creating and destroying a variable every time when it comes into and goes out of scope, static is initialized only once and remains into existence till the end of program. A static variable can either be local or global depending upon the place of declaration.

```
int* f1();
int main()
{
int *res = f1();
printf("Res is %p %d\n",res,*res);          //same address in all three outputs. Then 11 res = f1();
printf("Res is %p %d\n",res,*res);          // 12 res = f1();
printf("Res is %p %d\n",res,*res);          // 13 return 0;
}
int* f1()
{
static int a= 10;// memory is shared. This line is ignored after first function call
 a++;
return &a;     // valid because life time of static variable is through out the file execution
}
```

register: Registers are faster than memory to access, so the variables which are most frequently used in a C program can be put in registers using register keyword. The keyword register hints to compiler that a given variable can be put in a register. It's compiler's choice to put it in a register or not. Generally, compilers themselves do optimizations and put the variables in register. If a free register is not available, these are then stored in the memory only. If & operator is used with a register variable then compiler may give an error or warning (depending upon the compiler used), because when a variable is a register, it may be stored in a register instead of memory and accessing address of a register is invalid.

```
int main()
{       register int i = 10;        int* a = &i;
        printf("%d", *a);           getchar();        return 0;      }
```

b) Write a function definition to make this client code work without any error. The function must find the product of all the arguments passed to it except the first one. The first argument signifies the number of items following it to perform multiplication. 4 M

```
#include <stdarg.h>
int main()
{
        printf("product is %d\n",find_product(4,1,0,3,4));
        printf("product is %d\n",find_product(3,1,3,4));
        printf("product is %d\n",find_product(2,2,9));
        return 0;
}
```

Solution:

```
int find_product(int n,...) // ... ellipsis symbol   1 mark
{
        int s = 1;
        va_list va;    // 1 mark
        va_start(va,n);     // 1 mark
        for(int i  = 0;i<n;i++)
        {
                s*=va_arg(va,int);          // 1 mark
        }
        va_end(va);
        return s;
}
```

c) i) If the C code is run using the command **a.exe "12_14_16"**, then what is the output of below statements when executed separately? argc is the number of command line arguments in the command line passed with the executable and argv is an array of character pointers 6 M

```
printf("%d\t",strlen(argv[0]));
printf("%d\t",argc);
printf("%s\n",argv[1])
```

ii) What is the output of below code?

```
#include<stdio.h>
#define sqr(y) (y*y)
#define cube(x) sqr(x)*x
int main()
{       int x = 3;          printf("%d\t",cube(2+2));
        printf("%d\t",sqr(x));      printf("%d\n",sqr(4));                return 0;      }
```

Solution: 1 mark each

i) 5 2 12_14_16

ii) 18 9 16