Assigment-1

Data Structures (Code: CSO-102)

1. **Write C statements (corresponding to a program segment) for the following:**
   - **(a) Declare a variable *x* of type float and initialize it to 100.**
   - **(b) Declare *a* and *b* of type int.**
   - **(c) Read *a* and *b* from the user.**
   - **(d) Compute *a* divided by *b* with proper type cast so that no information is lost, and store the result in *x*.**
   - **(e) Print the value of *x*.**

Ans 1. (a) float x = 100;

(b) int a, b;

(c) scanf("%d %d", a, b);

(d) float x = (float) a / b;

(e) printf("%d", x);

2. **What will be printed when the following program statements / segments will execute?**

(a)

```
int x;
float y, z;
x = 10/3;
y = x/3;
z = x+y;
printf ("y = %f, z=%f", y, z)
;
```

(b)

```
#define CALC(X) (X*X)
int main() {
    int a, b=5;
   a = CALC(b+2);
    printf("\n a= %d b=%d", a,b);
}
```

(c)

```
int a=10;
if(a>=5)
     a=a+3;
else
     a=a+2;
printf("\n a=%d ",a);
```

(d)

```
int i,a[10];
a[0]=0;
for (i=1; i<10; i++)
a[i]=a[i-1]+i;
printf("\n val1=%d val2=%d",a[4],
a[9]);
```

Ans 2. (a) y = 1.000000, z=4.000000
(b) a= 17 b=5
(c) a=13
(d) val1=10 val2=45

3. **An integer is a perfect square if its square root is also an integer. Write a full program in C to print all the odd perfect squares between 1 and *N*, where *N* is read from the user.**

Ans 3.

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

int main ()
{
   int N;
   printf("Enter the value of N: ");
   scanf("%d", &N);
   printf("Odd Perfect squares between 1 and %d are: \n", N);
   int i = 1;
   while (i*i <= N)
   {
      printf("%d\n", i*i);
      i += 2;
   }
}
```

4. **Suppose, we define a node in the list in the usual way as:**

```
typedef struct _node {
        int data;
        struct _node *next;
} node;
```

   (a) **Let Head be a pointer to the first node in a sorted linked list. Head is NULL if the list is empty. We plan to insert a data value V in the list such that the list continues to remain sorted after the insertion. The value V to be inserted may or may not be already present in the list. Write down the function to above said problem and function should return a pointer to the first node in the modified list.**

   (b) **Assume that you are given a sorted linked list with possible duplicate data items stored in consecutive nodes. Write the function that removes all duplicate values (that is, if a data value is present multiple times, the function will retain only one instance of the data). The function returns a pointer to the updated list having no duplicate items.**

Ans 4 (a)

```
// Assuming it to be sorted in ascending order
node *insert_asc(node *head, int v) {
    node *newnode = (node *) malloc (sizeof(node));
    newnode -> data = v;
    newnode -> next = NULL;

    if(head == NULL || head -> data >= v) {
        newnode -> next = head;
        return newnode;
    }
    node *p = head, *prev;
    while (p != NULL && p->data < v) {
        prev = p;
        p = p->next;
    }
    prev -> next = newnode;
    newnode -> next = p;
    return head;
}
```

(b)

```
// Assuming it to be sorted in ascending order
node *removeDup (node *head) {

    if (head == NULL)
            return head;
    node *p = head->next, *prev = head;
    while (p != NULL) {
        if(prev -> data == p->data) {
            node *temp = p;
            p = p -> next;
            free(temp);
        }
        else {
            prev -> next = p;
            prev = p;
            p = p -> next;
        }

    }
    return head;
}
```

5. **Suppose a circular queue of capacity (n – 1) elements is implemented with an array of n elements. Assume that the insertion and deletion operation are carried out using REAR and FRONT as array index variables, respectively. Initially, REAR = FRONT = 0. Write the conditions to detect queue is full and queue is empty.**

Ans 5.

rear = Write

front = Read

full: (REAR+1) mod n==FRONT
empty: REAR==FRONT

6. **Suppose two stacks are given S1 and S2. Write the insert and delete operations on queue Q using stacks S1 and S2.**
   void insert(Q, x) {…}
   delete(Q){ …}

Ans 6.

```
void insert (Q, x) {
   push (S1, x);
}

void delete (Q) {
   if (stack-empty (S2))
      if(stack-empty (S1)) {
         printf("Q is empty");
         return;
      }
      else while (!(stack-empty(S1))) {
         x = pop (S1);
         push (S2, x);
      }
   x = pop (S2);
}
```