Question 1:

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node *next;
};
struct node *start = NULL;
void createNodes(int n){
   struct node *newNode;
   printf("The data in the reversed order is: \n");
   for(int i=0; i<n; i++){
   newNode = (struct node*)malloc(sizeof(struct node));
      if(start == NULL){
         printf("\nEnter the data: ");
         scanf("%d",&newNode->data);
         start = newNode;
         newNode->next = NULL;
      }else{
         printf("\nEnter the data: ");
         scanf("%d",&newNode->data);
         newNode->next = start;
         start = newNode;
      }
   }
}
void displayReverse(){
   struct node *temp;
   temp = start;
   printf("\n");
   while(temp != NULL){
      printf("%d\t",temp->data);
      temp = temp->next;
   }
}
int main(){
   int n;
   printf("Enter the number of nodes required: ");
   scanf("%d",&n);
   createNodes(n);
   displayReverse();
   return 0;
}
```

Question 2:

```
#include <stdio.h>
int countDuplicates(int arr[], int size) {
    int count = 0;
    for(int i=0; i<size; i++){
        if(arr[i] == -1){
            continue;
        }
        int duplicates = 0;
        for(int j=i+1; j<size; j++){
            if(arr[i] == arr[j]){
                duplicates++;
                arr[j] = -1;
            }
        }
        if(duplicates>0){
            count++;
        }
    }
    return count;
}
int main() {
    int arr[] = {1, 2, 3, 4, 1, 6, 1, 6, 4, 10};
    int size = sizeof(arr) / sizeof(arr[0]);
    int duplicateCount = countDuplicates(arr, size);
    printf("Total number of duplicate elements: %d\n", duplicateCount);
    return 0;
}
```

Question 3:

```
#include<stdio.h>
void printUniqueElements(int arr[], int size){
    for(int i=0; i<size; i++){
        int count = 0;
        if(arr[i] == -1){
            continue;
        }
        for(int j=i+1; j<size; j++){
            if(arr[i] == arr[j]){
                count++;
                arr[j] = -1;
            }
        }
        if(count == 0){
            printf("%d\t",arr[i]);
        }
    }
    printf("\n");
}
int main(){
    int arr[] = {3,2,2,5,5,6,3,7,8,6};
    int size = sizeof(arr)/sizeof(arr[0]);
    printf("The unique elements in the array: \n");
    printUniqueElements(arr,size);
}
```

Question 4:

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
#define MAX 20
struct stack{
    int arr[MAX];
    int top;
};
void push(struct stack *s, int element){
    if(s->top == MAX-1){
        printf("Overflow!!\n");
        exit(1);
    }
    s->arr[++s->top] = element;
}
int pop(struct stack *s){
    if(s->top == -1){
        printf("Underflow!!\n");
        exit(1);
    }
    return s->arr[s->top--];
}
int postfixEvaluation(char *expression){
    struct stack s;
    s.top = -1;
    for(int i=0; expression[i] != '\0'; i++){
        if(isdigit(expression[i])){
            push(&s, expression[i]-'0');
        }else{
            int op2 = pop(&s);
            int op1 = pop(&s);
            switch(expression[i]){
                case '+':
                    push(&s, op1+op2);
                    break;
                case '-':
                    push(&s, op1-op2);
                    break;
                case '*':
                    push(&s, op1*op2);
                    break;
                case '/':
                    push(&s, op1/op2);
```

```
                break;
            case '%':
                push(&s, op1%op2);
                break;
            default:
                printf("Invalid expression!!\n");
                exit(1);
                break;
        }
      }
   }
   return pop(&s);
}
int main(){
   char expression[30];
   printf("Enter a valid postfix expression: ");
   scanf("%s",expression);
   int res = postfixEvaluation(expression);
   printf("Result is : %d\n",res);
   return 0;
}
```

Question 5:

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   int pno;
   struct node *next;
};
struct node *start = NULL;
void insertion(int data, int pno){
   struct node *newNode, *temp;
   newNode = (struct node*)malloc(sizeof(struct node));
   newNode->data = data;
   newNode->pno = pno;
   if(start == NULL){
      temp = start = newNode;
      newNode->next = NULL;
   }else{
      temp = start;
      if(pno < temp->pno){
         newNode->next = start;
         start = newNode;
      }else{
         while(temp->next != NULL && pno>=temp->next->pno){
            temp = temp->next;
         }
         newNode->next = temp->next;
         temp->next = newNode;
      }
   }
}
void deletion(){
   struct node *temp = start;
   start = start->next;
   free(temp);
}
void display(){
   printf("Elements are:\n");
   for(struct node *i = start;i != NULL;i = i->next){
      printf("%d\t",i->data);
   }
   printf("\n");
}
void main(){
```

```
    int choice, data, priority;
    while(1){
        printf("The operations available are:\n");
        printf("1.insertion\n2.deletion\n3.display\n4.exit\n");
        printf("Enter your choice: ");
        scanf("%d",&choice);
        switch(choice){
            case 1:
                printf("\nEnter your data: ");
                scanf("%d",&data);
                    printf("\nEnter the valid priority: ");
                scanf("%d",&priority);
                insertion(data, priority);
                break;
            case 2:
                deletion();
                break;
            case 3:
                display();
                break;
            case 4:
                 exit(1);
                 break;
            default:
                printf("\nInvalid selection!!\n");
                break;
        }
    }
}
```